Maritime Analytics Prototype

Final Development Report

Oculus Info Inc and **Salience Analytics Inc**

Prepared by:
Eric Hall
Oculus Info Inc.

Neil Bozowsky
Oculus Info Inc.

Michael Davenport
Salience Analytics Inc.

William Wright
Oculus Info Inc.

Contract Number W7701-13-5425
April 2014

Contract Scientific Authority: Valérie Lavigne
(418) 844-4000 x 4114
DRDC – Valcartier Research Centre

Oculus Info Inc.
2 Berkeley Street, Suite 600
Toronto, ON, Canada, M5A 4J5
Tel: 416-203-3003
www.oculusinfo.com

Salience Analytics Inc.
87 West 41 Ave
Vancouver, BC, Canada, V5Y 2R8
Tel: 604-790-3771
www.SalienceAnalytics.ca

DRDC-RDDC-2014-C325

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

© 2014 Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence

© 2014 Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale

Abstract

This report documents the implementation of a Maritime Visual Analytics Prototype (MVAP), a test-bed and showcase for novel visual analytics strategies that will in the future support maritime situation awareness, anomaly detection, and detailed analysis of a Vessel of Interest (VOI). MVAP uses a Service Oriented Architecture, with reuse of services from DRDC Valcartier's Intelligence Science and Technology Integration Platform (ISTIP).

The development took place in two Phases from January to August of 2012, a third Phase from October of 2012 to June of 2013, and a Network Analysis Capabilities amendment from January to April of 2014. In Phase 1, four novel widgets or "apps" were developed. The widgets are focused, compact applications inspired by the style of an iPhone app. The "Vessel Summary Cards" deliver a unique visual representation of each ship, suitable for rapid visual browsing and data drill-down. The "Record Browser" widget allows analysts to rapidly shuffle through the Vessel Summary Cards to gain a visual impression of their assigned set of vessels without having to read all the details. The "Timeline" widget can show timelines of multiple ships, offset if necessary to reveal recurrent patterns of behavior. The "Magnets Grid" widget uses a dynamic scatterplot to sort events and objects based on their multi-dimensional properties. These widgets are supported by an "Analysis Set Manager" service that allows users to create and modify groups of ships and pass those groups between widgets.

Additionally, work was undertaken to analyze and improve upon the underlying web development framework known as the Widget Application Shell (WAS). Significant additional features implemented include the ability to save and restore workspace state, integration of the open source authentication and access management platform OpenAM, support for multiple instances of the same type of widget and support for installation specific configuration files to control both client and server behavior.

In Phase 2, the "Map and Timeline" widget was developed to allow the user to interactively view time-varying vessel positional data and vessel contact data. "Close Encounters Popups" display vessel tracks that have been in close proximity, and "Route Ribbons" highlight vessel deviation from an optimal path between two points.

Capabilities developed in Phase 3 included extending MVAP to improve the ability to deploy the visualizations as part of a shared collaborative workspace, and integrating artificial reasoning capabilities to manage smart sets of vessels. Standalone versions of all widgets were created that can be used outside of the WAS workspace, and selected features that were out of scope for MVAP development Phases 1 and 2 were added to the prototype.

As part of the Network Analysis Capabilities amendment MVAP widgets were modified to support network graph analysis, and a new Graph Analyzer widget was developed.

Demonstration scenarios were developed in Phases 1 and 2, and data was assembled in a PostgreSQL database, to support the demonstrations. An installation guide and demonstration instructions are also provided in this document.

Résumé

Ce rapport documente l'implémentation d'un Prototype d'analytique visuelle maritime (PAVM), un banc d'essai et une opportunité de présenter de nouvelles stratégies d'analyse visuelle qui pourront supporter la connaissance de la situation maritime, la détection d'anomalies, et l'analyse détaillée d'un Navire d'intérêt (NDI). Ce prototype utilise une Architecture axée sur les services (AAS), et réutilise la Plateforme d'intégration de la science et de la technologie du renseignement (PISTR) de RDDC Valcartier.

Le développement a été réalisé en deux phases de janvier à août 2012, lors d'une troisième phase d'octobre 2012 à juin 2013, et dans le cadre de l'amendement « Network Analysis Capabilities » de janvier à avril 2014. Dans la phase 1, quatre nouveaux widgets ou « apps » ont été développés. Ces widgets sont des applications compactes et concentrées inspirées du style présenté par les applications iPhone. Le widget «Vessel Summary Card» offre une représentation visuelle unique de chaque navire, permettant une exploration visuelle rapide avec *drill-down*. Le widget «Record Browser» permet aux analystes de parcourir rapidement les Vessel Summary Cards pour obtenir une impression visuelle de l'ensemble de navires leur étant assigné sans avoir à lire tous les détails. Le widget «Timeline» peut montrer des lignes de temps de plusieurs navires, alignées afin de révéler des tendances ou des comportements récurrents. Le widget «Magnets Grid» utilise un diagramme de dispersion dynamique pour classer les évènements et les objets en fonction de leurs propriétés multidimensionnelles. Ces widgets sont pris en charge par le service «Analysis Set Manager» qui permet aux utilisateurs de créer et modifier des groupes de navires et de transmettre ces groupes entre les widgets.

De plus, un travail d'analyse et d'amélioration du cadriciel de développement web nommé le *Widget Application Shell* (WAS) supportant le PAVM a été entrepris. D'importantes fonctionnalités supplémentaires mises en œuvre par Oculus comprennent la possibilité d'enregistrer et restaurer un état de l'espace de travail, l'intégration de la plate-forme *open source* de l'authentification et de la gestion de l'accès OpenAM, le support pour plusieurs instances du même type de widget et le support des fichiers d'installation de configuration spécifiques pour contrôler à la fois le comportement du client et du serveur.

Dans la phase 2, le widget « Map and Timeline » a été développé afin de permettre à l'utilisateur de visualiser de manière interactive des données variant temporellement concernant les positions et les contacts des navires. « Close Encounters Popups » montre les trajectoires des navires qui se sont trouvées à proximité l'une de l'autre et « Route Ribbons » met en évidence la déviation de la trajectoire d'un navire par rapport à la trajectoire optimale entre deux points.

Les capacités développées au cours de la phase 3 comportent l'extension du PAVM afin d'améliorer la facilité de déployer les visualisations dans un espace de travail collaboratif, et l'intégration d'une capacité de raisonnement artificiel pour la gestion de listes de navires intelligentes. Des versions autonomes ont été créées pour tous les widgets afin de pouvoir les utiliser à l'extérieur de l'environnement WAS, et certaines fonctionnalités qui sortaient du cadre de travail des phases 1 et 2 de développement du PAVM ont été ajoutées au prototype. Des scénarios de démonstration ont été élaborés pour les phases 1 et 2, et les données ont été assemblées dans une base de données PostgreSQL pour soutenir la démonstration. Un guide d'installation et des instructions pour la démonstration sont également fournis dans le présent document.

Dans le cadre de l'amendement « Network Analysis Capabilities » les widgets MVAP ont été modifiés afin de supporter l'analyse de graphes de réseaux sociaux et un nouveau widget « Graph Analyzer » a été développé.

Executive summary

Maritime Visual Analytics Prototype: Development Report

Eric Hall; Neil Bozowsky; Michael Davenport; William Wright; Defence R&D Canada – Valcartier; April 2014; Unclassified.

Introduction or background:

This report documents development Phases 1, 2 and 3 of a Maritime Visual Analytics Prototype (MVAP) test-bed implementation, and additional Network Analysis Capabilities development. This work was done under Advanced Research Project (ARP) 11jm (previously 11hm) "Maritime Domain Analysis through Collaborative and Interactive Visualization." It builds on an earlier requirements analysis, literature and product survey, plus design and architecture studies.

This research specifically examines how Visual Analytics can help surveillance agencies achieve better maritime domain awareness by offering cognitively rich representations, information filtering and clutter reduction capabilities, multimodal interactions, and teamwork collaboration services. To avoid reproducing already well-established and widely used software systems, the focus is on developing compact apps (a commonly-used abbreviation for "applications") or widgets that might be incrementally added to existing surveillance systems. For the project, Oculus was responsible for project management, technical architecture, user experience design refinement, software development and demo scenarios. Salience was responsible for subject matter expertise, functional design refinement and demo scenarios.

Results:

MVAP was implemented using a Service Oriented Architecture, with reuse of services from DRDC Valcartier's *Intelligence Science and Technology Integration Platform* (ISTIP). Code is written in JavaScript and Java, using libraries such as *Google Web Toolkit*, *jQuery*, and Oculus's *Aperture*. Scenarios are represented using hand-crafted data as well as data gleaned from websites, and are stored for MVAP in a PostgreSQL database.

The following widgets were developed:

- Vessel Summary Card: delivers a unique visual representation of each ship suitable for rapid visual browsing and with quick links to drill-down information.
- Record Browser: allows analysts to rapidly shuffle through the Vessel Summary Cards to gain a visual impression of their assigned set of vessels without having to read all the details.
- Timeline: shows relevant events as icons on multiple parallel timelines. Timelines can be time-shifted relative to each other to find repeating time patterns and anomalies.
- Magnets Grid: dynamic scatter plot visualization to detect patterns in large collections of objects characterized by various properties.
- Word Clouds: text-based visualizations emphasizing words that are most relevant.
- Map and Timeline: allows the user to interactively view time-varying vessel positional data and vessel contact data. "Close Encounters Popups" display vessel tracks that have been in

close proximity, and “Route Ribbons” highlight vessel deviation from an optimal path between two points.

- Graph Analyzer: allows the user to interactively view links and relations between archives.

These widgets are supported by an “Analysis Set Manager” service that allows users to select groups of ships and pass those groups between widgets.

Additionally, work was undertaken to analyze and improve upon the underlying web development framework known as the Widget Application Shell (WAS). Significant additional features implemented include the ability to save and restore workspace state, integration of the open source authentication and access management platform OpenAM, support for multiple instances of the same type of widget and support for installation specific configuration files to control both client and server behaviour.

Capabilities developed in Phase 3 included extending MVAP to improve the ability to deploy the visualizations as part of a shared collaborative workspace, and integrating artificial reasoning capabilities to manage smart sets of vessels. Standalone versions of all widgets were created that can be used outside of the WAS workspace, and selected features that were out of scope for MVAP development Phases 1 and 2 were added to the prototype.

Significance:

Previous studies have shown that Visual Analytics is well-positioned to help analysts achieve better maritime domain awareness by offering cognitively rich representations, information filtering and clutter reduction capabilities, multimodal interactions and teamwork collaboration services. This will be particularly relevant to the member agencies of the Maritime Surveillance Operations Centres (MSOCs), including the Royal Canadian Navy.

The MVAP will act as a showcase for relevant novel visual analytics strategies that will in the future support maritime situation awareness, anomaly detection, and detailed analysis of a Vessel of Interest.

Table of Contents

Abstract ……. i
Résumé …. ii
Executive summary iv
Table of Contents vi
List of figures xi
1 Background and Functional Context 1
1.1 Objectives 1
1.2 Context and Background 1
1.2.1 Related DRDC Projects 1
1.2.2 Previous Work under ARP 11jm/11hm 2
1.2.3 Maritime Domain Awareness 2
1.2.4 Maritime Surveillance Operational Centers 2
1.2.5 Visual Analytics 4
1.2.6 App-Based Design 4
2 Software Architecture 5
2.1 Source Code Structure 5
2.2 Eclipse Project Source Code Deliverables 5
2.2.1 MVAP Data Import Scripts 5
2.3 Browser Based Database App 6
2.4 GWT and REST Interfaces to MVAP Database Objects 7
2.5 Database Structure for Archives and Analysis Sets 7
2.6 Database Structure for Archives and Links 8
2.7 REST Data Objects 9
2.8 REST URLs 12
2.9 GPW REST calls 20
2.10 GWT and JavaScript Integration 21
2.11 jQuery JavaScript Library 21
2.12 Aperture JavaScript Library 21
2.13 Generic Widget Architecture 21
2.13.1 Visual Summary Card 22
2.13.2 Magnets Grid 26
2.13.3 Timeline 28
2.13.4 Map and Timeline 29
2.14 GPW Data Analysis 30
2.14.1 Database structure 31
2.14.2 GPW preprocessing 31
2.14.3 GPW Map and Timeline Widget 32
3 Design of the Apps 33
3.1 MVAP Functionality 33

3.1.1 Analysis Set Manager 33
3.1.1.1 Add, Update, Remove 33
3.1.1.2 Right click menu 35
3.1.1.3 ISTIP Artificial Reasoning Sets and Social Network Analysis created Sets 35
3.1.1.4 Other Functions 36
3.1.2 Vessel Summary Card 36
3.1.3 Person Summary Card 38
3.1.4 Record Browser 39
3.1.4.1 Comparison Card 41
3.1.4.2 Export Selected Archives 41
3.1.4.3 Mark as VOI 42
3.1.4.4 Reset Selection 43
3.1.5 Timeline 44
3.1.5.1 Timeline Controls: Zoom, Move, Lock & Unlock 45
3.1.5.2 Timeline Controls: Filter and Jump to Date 46
3.1.5.3 Remove Ships 47
3.1.5.4 Export Subset 47
3.1.6 Magnets Grid 47
3.1.6.1 Attribute Customization 48
3.1.6.2 Calculated Attributes 49
3.1.6.3 Magnets 49
3.1.6.4 Dust 50
3.1.6.5 Axes 51
3.1.6.6 Other Controls: Ship information, Select ship groups, Shake Magnets 53
3.1.6.7 Standalone Magnets Grid 53
3.1.7 Map and Timeline 54
3.1.7.1 Close Encounters 56
3.1.7.2 Route Ribbons 57
3.1.7.3 Map Controls 58
3.1.8 Graph Analyzer 58
3.1.8.1 Links Tab 59
3.1.8.2 Analysis Tab 59
3.1.8.3 Search Tab 61
3.1.8.4 Disconnected Components 61
3.1.8.5 Node Selection 62
3.1.8.6 Context Menu 62
3.1.8.7 Toolbar 62
3.1.8.8 Importing Graph Data 63
3.1.9 Standalone deployment 64
3.1.10 GPW Map and Timeline 65
3.2 Design Feedback Not Implemented 67
3.2.1 Analysis Set Manager 67

3.2.2 Vessel Summary Card 67
3.2.3 Timeline 68
3.2.4 Magnets Grid 68
3.2.5 Map and Timeline 68
3.2.6 Pop-ups 68
3.2.7 Workspace 68
3.2.8 Graph Analyzer 68
4 Widget Application Shell (WAS) Development 69
4.1 WAS Framework: Context, Objectives and Scope 69
4.1.1 Context 69
4.1.2 Objectives 70
4.1.3 Deliverables 70
4.1.4 Framework Alternatives 70
4.2 Oculus Development Work 70
4.2.1 Secure User Login 70
4.2.2 Client/Server WAS Settings & Configurable Workspaces 71
4.2.3 Persistent Workspaces 71
4.3 WAS Limitations 71
4.3.1 Google Web Toolkit (GWT) Limitations 72
4.3.2 SmartGWT Limitations 72
4.3.3 WAS Issues 72
4.3.3.1 Tightly Coupled Compilation 72
4.3.3.2 User Stories and Workflow Design 73
4.3.3.3 Facilitating Widget UI Layout 73
4.4 WAS Alternative: Ozone Widget Framework and Synapse 73
4.4.1 Ozone Widget Framework (OWF) 73
4.4.2 Synapse 74
4.5 Future Work 74
4.5.1 Further WAS Improvements 74
4.5.1.1 User Sharing 74
4.5.1.2 Data Model Architecture Design 75
4.5.1.3 Workspace Administration 75
4.5.1.4 WAS Replacement 75
4.6 WAS Developer Guide 76
4.6.1 Client WAS Settings & Configurable Workspaces 76
4.6.2 Server WAS Settings 76
4.6.3 Persistent Workspaces 76
4.6.4 Tutorial: Saving Data 77
4.6.5 Tutorial: Loading and Re-creating Objects 79
4.7 OpenAM Deployment 81
4.7.1 Server Pre-deployment Configuration 81
4.7.2 Server Installation 82
4.7.3 Server Configuration 82

4.7.4 Protecting a Web Application....83
5 Demonstration Scenarios....87
5.1 Data Mapping to the Maritime Domain....87
5.2 IRISL Scenario....89
5.2.1 Background: IRISL....90
5.2.2 Description of the Scenario....90
5.2.2.1 Part A: Loitering Vessels....90
5.2.2.2 Part B: Jacardian Vessels....91
5.3 Development Phase 1 and 2 Project Conclusion Demonstration Scenario....92
5.3.1 Part A: Launch the demonstration....92
5.3.2 Part B: View Vessels and Sets in the Analysis Set Manager....92
5.3.3 Part C: Scan Vessels in the Record Browser....93
5.3.4 Part D: Launch a Map and Fetch Vessels for a Time and Region of Interest....93
5.3.5 Part E: Analyze Vessel Motion and Encounters....93
5.3.6 Part F: Create an Analysis Set....94
5.3.7 Part G: Analyse the Set in Magnets Grid....94
5.3.8 Part H: Mark the vessels as VOIs....94
5.3.9 Part H: Analyse VOIs in the Timeline....94
5.4 Additional Demonstration Scenario....95
5.4.1 Part A: Scan the Cards....95
5.4.2 Part B: View the Tracks....95
5.4.3 Part C: Route Ribbons Anomaly Detection....95
5.4.4 Part D: Pop-Up Anomaly Detection....96
5.4.5 Part E: Detailed Analysis of a VOI using Timelines....96
5.4.6 Part F: Magnets Grid Anomaly Detection....96
5.5 Demonstration Videos....97
5.6 Demonstration Setup Guide....97
5.6.1 Install Apache Tomcat....97
5.6.2 Install PostgreSQL....98
5.6.3 Download Prototype files & Copy ‘war’ file into /webapps....99
5.6.4 Restore Database....99
5.6.5 Run the Prototype....101
5.6.5.1 Start Tomcat Server....101
5.6.5.2 Connect to Web Application....102
5.6.5.3 Shutdown Tomcat Server....104
6 Test Procedure....105
6.1 Framework and Workspace....105
6.2 Analysis Set Manager....105
6.3 Record Browser....106
6.4 Timeline....107
6.5 Magnets Grid....108
6.6 Map and Timeline....109

6.7 Standalone 111
6.8 Graph Analyzer 111
References 117
Bibliography 118
List of symbols/abbreviations/acronyms/initialisms 121

List of figures

Figure 1: Database Structure for Archives and Analysis Sets 8
Figure 2: Database Structure for Archives Links 9
Figure 3: Database Structure for GPW Data Analysis 31
Figure 4: Analysis Set Manager 33
Figure 5: Analysis Set Manager: Add Ship Group & Add Ship buttons 34
Figure 6: Select Archives Dialog 34
Figure 7: Analysis Set Manager Context Menu 35
Figure 8: Autogenerated Set with Artificial Reasoning 36
Figure 9: Analysis Set Manager: Widget Access 36
Figure 10: Vessel Summary Card 37
Figure 11: Vessel Summary Card: Flip Side 38
Figure 12: Person Summary Card 39
Figure 13: Record Browser 40
Figure 14: Record Browser for Social Network Analysis 40
Figure 15: Record Browser: Export Ships 41
Figure 16: Export archives: Select analysis set, Enter new analysis set name 42
Figure 17: Record Browser: Mark as VOI 43
Figure 18: Record Browser: Reset Selection 43
Figure 19: Timeline 44
Figure 20: Timeline: Event Overview 44
Figure 21: Timeline for Social Network Analysis: Event Overview 45
Figure 22: Timeline Controls: Zoom, Move, Lock & Unlock 45
Figure 23: Timeline Controls: Filter and Jump to Date 46
Figure 24: Timeline Controls: Set Timeline Filter 46
Figure 25: Timeline: Row Controls 47
Figure 26: Magnets Grid 48
Figure 27: Magnets Grid: Attribute Creation and Removal 49
Figure 28: Magnets Grid: Calculated Attributes 49
Figure 29: Magnets Grid: Magnets 50
Figure 30: Magnets Grid: Magnet Controls 50
Figure 31: Magnets Grid: Dust 51

Figure 32: Magnets Grid: Axes 52
Figure 33: Magnets Grid: X and Y Axes 53
Figure 34: Standalone Magnets Grid 54
Figure 35: Map and Timeline 55
Figure 36: Standalone Map and Timeline 56
Figure 37: Close Encounters 57
Figure 38: Graph Analyzer 58
Figure 39: Interactive Legend 59
Figure 40: Standard and Analytic Links 60
Figure 41: Analytic Link Threshold 60
Figure 42: Graph Searching 61
Figure 43: Disconnected Components 61
Figure 44: Node Context Menu 62
Figure 45: Graph Analyzer Toolbar 62
Figure 46: Graph Import Dialog 63
Figure 47: Autogenerated Analysis Sets from Importing Graph Data 64
Figure 48: Standalone MVAP Main Page 64
Figure 49: GPW Map and Timeline 65
Figure 50: GPW Map and Timeline Context Menu 66
Figure 51: GPW Encounters 67
Figure 52: OpenAM: Creating a New Rule 84
Figure 53: OpenAM: Allowing Access for Users 84
Figure 54: OpenAM: Login Page 86
Figure 55: Key Elements of Maritime Domain Situation Assessment 88
Figure 56: Primary Sources of Data for the MVAP Apps 89
Figure 57: The PostgreSQL Administration Tool 100
Figure 58: Creating a New Database 100
Figure 59: Restoring a Database Backup 101

1 Background and Functional Context

This is the Final Development Report, documenting Phase 1, Phase 2, Phase 3 and Network Analysis Capability amendment development, for the Maritime Visual Analytics Prototype project which spanned two consecutive contracts (PWGSC).

1.1 Objectives

The objective of this project is to develop the Maritime Visual Analytics Prototype (MVAP). The MVAP will demonstrate the use of Visual Analytics science and technology for Visual Analytics based Detection of Anomalies (VADA).

This work is part of Advanced Research Project (ARP) 11jm (previously 11hm), titled "Maritime Domain Analysis through Collaborative and Interactive Visualization" project, also known as "Visual Analytics for Maritime Domain Awareness" (VAMDA). This project is exploring and demonstrating how VA and collaborative technologies can:

- Help Canadian Armed Forces (CAF) rapidly grasp and identify key information elements and thus gain insight into the current maritime situation;
- Improve the visualization of the recognized maritime picture;
- Enable better comprehension of a situation and how it could develop;
- Provide visibility into what is not known (data gaps and uncertainty);
- Support detection, alerting, and visualization of anomalies;
- Enable collaborative team work.

1.2 Context and Background

The prototype developed in this project sits on an extensive foundation of background research done by Defence Research and Development Canada (DRDC), as outlined in Sections 1.2.1 and 1.2.2. Sections 1.2.3 through 1.2.5 then review key insights from those projects that are influential in the current project.

1.2.1 Related DRDC Projects

This project builds on foundational work done by DRDC in the following fields:

- Information visualization and management for enhanced domain awareness in maritime security (Project 11he).
- Collaborative Knowledge Exploitation for Maritime Domain Awareness (Project 11hg).
- Multi-hypothesis Link Analysis for Anomaly Detection in the Maritime Domain (Project 11hk).
- Social Network Analysis in Counter-Insurgency Context (Project 05do).

1.2.2 Previous Work under ARP 11jm/11hm

This project is part of a series that have been funded under DRDC's "Maritime Domain Analysis through Collaborative and Interactive Visualization" project ARP 11jm (previously 11hm), also known as Design of a Maritime Visual Analytics Prototype (DMVAP). Previous contracts under this project include:

- Literature and product survey [1];
- MSOC (Maritime Surveillance Operational Centers) site visits and requirements analysis [2];
- Design study for the prototype [3];
- Architecture study for the prototype [4];
- Scenario descriptions and dataset requirements analysis [5].

1.2.3 Maritime Domain Awareness

Maritime domain operators/analysts around the world typically have a mandate to achieve Maritime Domain Awareness, which is defined as follows:

- "Maritime Domain Awareness (MDA) is the effective understanding of everything on, under, related to, adjacent to, or bordering a sea, ocean or other navigable waterway, including all maritime-related activities, infrastructure, people, cargo, vessels, or other conveyances." [6]

This mandate is based on the need to protect from attack, defend sovereignty, detect illegal activities, and support search and rescue activities. Operators and analysts maintain 24/7 watch over the oceans in support of the mandate. They do so by extracting and analyzing situational facts from a variety of sensor data streams and analysis tools.

In Canada, the Regional Joint Operations Centres (RJOCs), through the production and exploitation of the Recognized Maritime Picture (RMP), need to rapidly process a variety of sources of information in order to develop shared situational awareness (understand how a situation has developed and is expected to develop); rapidly develop shared understandings of the operational environment; and work in a collaborative setting. In modern operations, this becomes a challenging task because of:

- The large number and diversity of data, information and knowledge types and sources;
- Significant information overload;
- The incomplete / uncertain nature of the information; and
- The timeliness of the information and of the resulting RMP.

1.2.4 Maritime Surveillance Operational Centers

Responsibility for gathering Maritime information, and for responding to actionable information, is divided among the member agencies of the Maritime Surveillance Operational Centers (MSOCs). The identities and roles of these agencies can be summarized as follows:

- **Royal Canadian Navy and the Canadian Armed Forces:**

- **Surveillance Role:** host agency for the Halifax and Esquimalt MSOCs, lead agency for assembling the global Recognized Maritime Picture, operator of the CAF maritime surveillance aircraft, ships, and sensor systems, and primary point of contact for exchanging information with international military surveillance agencies. Maintains situational awareness throughout Canada's area of responsibility (which extends beyond Canadian territorial waters) and in selected regions around the world, such as around Canadian warships. CAF is not allowed to collect information on Canadian citizens.
- **Response Mandate:** primary responder for events in international waters, for events involving a foreign warship, and for any maritime act of war against Canada. Frequently requested to support other MSOC agencies (e.g. RCMP) for actions within their jurisdiction.

- **Royal Canadian Mounted Police (RCMP):**
  - **Surveillance Role:** host agency for the Great Lakes MSOC, able to collect some information on Canadian citizens.
  - **Response Mandate:** primary responder for law enforcement against ships in Canadian territorial waters, including the Great Lakes, Atlantic, Pacific, and Arctic oceans.
- **Department of Fisheries and Oceans (DFO) including the Canadian Coast Guard (CCG):**
  - **Surveillance Role:** CCG maintains situational awareness of ship movements in Canadian waters through its Marine Communications and Traffic Services (MCTS) operations centers. They collect call-in data from ships approaching Canadian ports, and track the locations of the ships using their network of shore-based radars and Automatic Identification System (AIS) receivers. DFO flies surveillance aircraft, primarily to monitor fishing zones. Both DFO and CCG operate ships that provide surveillance data. DFO has access to vessel tracking system (VTS) data from fishing boats in or near Canadian waters.
  - **Response Mandate:** CCG enforces conformance to shipping regulations. DFO enforces conformance to fishing regulations and restrictions. Both DFO and CCG can intercept and board ships suspected of infringing regulations.
- **Transport Canada (TC):**
  - **Surveillance Role:** oversees security at Canada's ports, detects marine pollution events, detects regulatory infractions, regulates commercial shipping in Canadian waters, regulates security at offshore facilities such as oil and natural gas drilling platforms.
  - **Response Mandate:** imposes regulations, collects evidence, fines or prosecutes offenders, educates and certifies mariners, shares information with other agencies.
- **Canada Border Services Agency (CBSA):**
  - **Surveillance Role:** collects information about cargo and passengers destined for Canada, before they leave a foreign port. They maintain a network of intelligence agents in Canada and abroad to detect threats to Canada's immigration, visitor, refugee and citizenship programs, and to detect contraband products.
  - **Response Mandate:** ability to block entry of persons or cargo into Canada. Provide intelligence reports that can lead to interventions by the RCN or RCMP.

1.2.5 Visual Analytics

A widely accepted summary of Visual Analytics is:

- "Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces. People use visual analytics tools and techniques to synthesize information and derive insight from massive, dynamic, ambiguous and often conflicting data; detect the expected and discover the unexpected; provide timely, defensible, and understandable assessments; and communicate assessment effectively for action" [7].

Visual analytics is thus a multidisciplinary field that includes the following focus areas:

- Analytical reasoning techniques that let users obtain deep insights that directly support assessment, planning, and decision making;
- Visual representations and interaction techniques that exploit the human eye's broad bandwidth pathway into the mind to let users see, explore, and understand large amounts of information simultaneously;
- Data representations and transformations that convert all types of conflicting and dynamic data in ways that support visualization and analysis;
- Techniques to support production, presentation, and dissemination of analytical results to communicate information in the appropriate context to a variety of audiences.

It is anticipated that visual analytics has the potential to significantly improve the Recognized Maritime Picture by offering cognitively rich representations, information filtering and clutter reduction capabilities, multimodal interactions and teamwork collaboration services, in order to provide better insight into information and increased situation awareness.

1.2.6 App-Based Design

The design study [5] outlined 23 potential applications for collaborative visual analytics of a Vessel of Interest (VOI), visualization of normal maritime behaviour, visual analytics for detection of anomalies, and visualization of the situational facts knowledge base. It recommended that these be developed and offered as compact add-ons or "apps" (in this document, also referred to as "widgets"), where "compact" means:

- **Transient:** Each app can "pop up" when needed and then hide away under an icon when not needed.
- **Self-Contained:** Apps can be added as a service without requiring changes to the operating system or other applications
- **Single-Purpose:** Users experience each app as doing one specific job.

This approach is motivated in part by CAF's long tradition of upgrading their maritime domain information technology through a series of incremental improvements, rather than through "system replacement" cycles. If VA capabilities are packaged as apps, end-users can pick and choose which are of most interest, and add them one at a time with minimal risk to their operations.

Prototypes of a selected subset of those apps have been built during the course of this project.

2 Software Architecture

2.1 Source Code Structure

The Maritime Visual Analytic Prototype (MVAP) is a set of JavaScript widgets which can be used independently or integrated into the Widget Application Shell (WAS) framework. DRDC provided Oculus with the source code for the WAS framework and incomplete source code for existing workspaces built using the WAS framework. This pre-existing source code included a number of Eclipse projects organized in a particular directory structure. In an effort to maximize reusability and minimize conflicts with existing projects, Oculus created the MVAP workspace and widgets in a manner consistent with the existing workspaces. It will thus be a reasonably simple exercise to re-integrate the MVAP workspace with the pre-existing workspaces.

2.2 Eclipse Project Source Code Deliverables

Oculus is delivering the workspace project "MVAPGWT" which is analogous to:

- MultiReasonerInferenceGWT,
- SituationalFactsManagementGWT,
- SituationalOntologyManagementGWT and
- SpatialFeaturesManagementGWT.

The MVAPGWT project includes an Ant script "build.xml" which can be used to build the MVAP workspace, copy the built files into the wasPrototype project (which may host additional workspaces), build the wasPrototype and deploy a war file.

In addition to the MVAPGWT project, Oculus is delivering the "MVAPUI" and "MVAPDB" projects. The MVAPUI project includes all of the JavaScript widgets plus REST-based client-server interfaces. MVAPUI can be built and deployed as a standalone web application or the components can be used within the WAS framework. MVAPDB includes all of the source code required to connect to a Postgres database and manipulate persistent MVAP data objects.

2.2.1 MVAP Data Import Scripts

The "MVAPDB" project also contains application code for populating the database by reading vessel data from a variety of sources such as Transport Canada, Ship Spotting, and Excel files. The applications do not include user friendly interfaces but can be accessed by developers. Developers can edit the java code to adapt it to new types of data import. The data used in the MVAP Phase 1 demonstration is included in the project.

In the source directory oculus.mvap.db.sources.vessels there is a class `VesselReader` which is used to read a set of vessels from the Transport Canada website and put the data into the archive service along with an image from the ship spotting website. To use `VesselReader`, go to the Transport Canada website at:

http://wwwapps.tc.gc.ca/Saf-Sec-Sur/4/vrqs-srib/m.aspx

Search for registration ID's using desired search criteria and then create a static array of these ID's in `VesselReader`. Execute the `readVesselListIntoDB()` function using the new array. It will delete all of the old vessels from the DB and add all of the new vessels. Results can be tested using the `VesselReader.readDB()` function which will display the vessel information from the database in a swing window.

The source file oculus.mvap.db.sources.events.EventBuilder has functions to clear all events from the database and to create random arrival and departure events with 24h and 96h reports for arrivals in Canada.

The source file oculus.mvap.db.sources.aset.ASetBuilder when executed will create a new analysis set called "All Vessels" which contains every vessel in the system.

The source file oculus.mvap.db.sources.xldata.NewVesselReader can be used to read data from a tab delimited text file plus an image directory into the database. Likewise, NewVesselDataReader and NewEventReader read tab delimited text into the vessel meta data and event tables respectively.

The source file oculus.mvap.db.sources.events.CountryReader is used to create country flag images using Wikipedia's database. It can be used in future to update these images. Modification will be required since the code uses a snippet of a Wikipedia website listing countries.

The source file oculus.mvap.dp.graph.GraphImport is used to import GraphDTO formatted data into the system. It can be run via the command line, or triggered by the user through the client user interface. This creates archives, meta data, autogenerated analysis sets, events (if applicable), and tracks (if applicable).

2.3 Browser Based Database App

The MVAP project includes a simple browser based user interface which can be used to populate the Postgres database. To use the DB interface, open a web browser and go to the address:

http://<MVAP_SERVER>/wasPrototype/test/dbview.html

The first tab, "Archive Browser", lists all archives in the database alphabetically ordered by name. Clicking the plus beside an archive will reveal more options.

- Tracks: Lists tracks in the output area with point count, start and end times.
- Data: Lists all archive meta data. Can be cut and pasted into the Set Data dialog for another vessel.
- Events: Lists all events associated with the archive and the event meta data.
- Image: Displays the image for the vessel. Right click and use "Copy Image URL" to transfer the image to another vessel.
- Set Image: Brings up a dialog which will read an image from a URL and write it to the database.

- Set Data: Brings up a dialog which accepts ";" delimited text to assign new meta data to the archive.
- Delete: Deletes the archive from the database.

The second tab, "Database Cleanup", provides options to clear the database. Clearing vessels also removes associated data, events, tracks, etc. Clearing vessels by prefix is useful in conjunction with adding vessels by prefix in the "File Reader" tab.

The third tab, "File Reader", allows different input files to be uploaded to the server and read into the database. It should be possible to drag a file over the "No file chosen" text. Tab delimited text files can be exported from excel.

- Upload Vessels: Creates new vessels in the database and assigns them IDs.
- Upload Meta: Assigns new meta data to existing vessels.
- Upload Events: Creates new events and associates them with existing vessels.
- Upload KML: Reads data files similar to those found in the ISTIP "DataInjector" project. New vessels will be created in the database with tracks found in the KML. A prefix can be specified so that the uploaded KML can be deleted later. An analysis set name can also be specified.
- Create All Vessel Set: Creates a new set called "All Vessels" containing every archive in the database.

2.4 GWT and REST Interfaces to MVAP Database Objects

Client-Server communication in the MVAP framework is achieved using a REST (Representational State Transfer) interface to the database using the “Jersey” Java REST API. REST is a very light-weight and simple web communication protocol that allows resources to be made available with URL identifiers linked to GET/PUT/DELETE functions and passing data with JSON (JavaScript Object Notation) or XML. Jersey automatically converts the JSON or XML input and output to or from Java objects and provides a simple mechanism to associate a URL with a given GET/PUT/DELETE function. For deployment with WAS and GWT, many of the service calls have been implemented using the GWT client-server protocol.

2.5 Database Structure for Archives and Analysis Sets

The database stores generic archives which represent vessels in the case of MVAP but could be used to store many types of data. See Figure 1. Each archive can have a name, id, image, set of meta data and set of events.

Archives can be stored in a tree-structure of analysis sets.

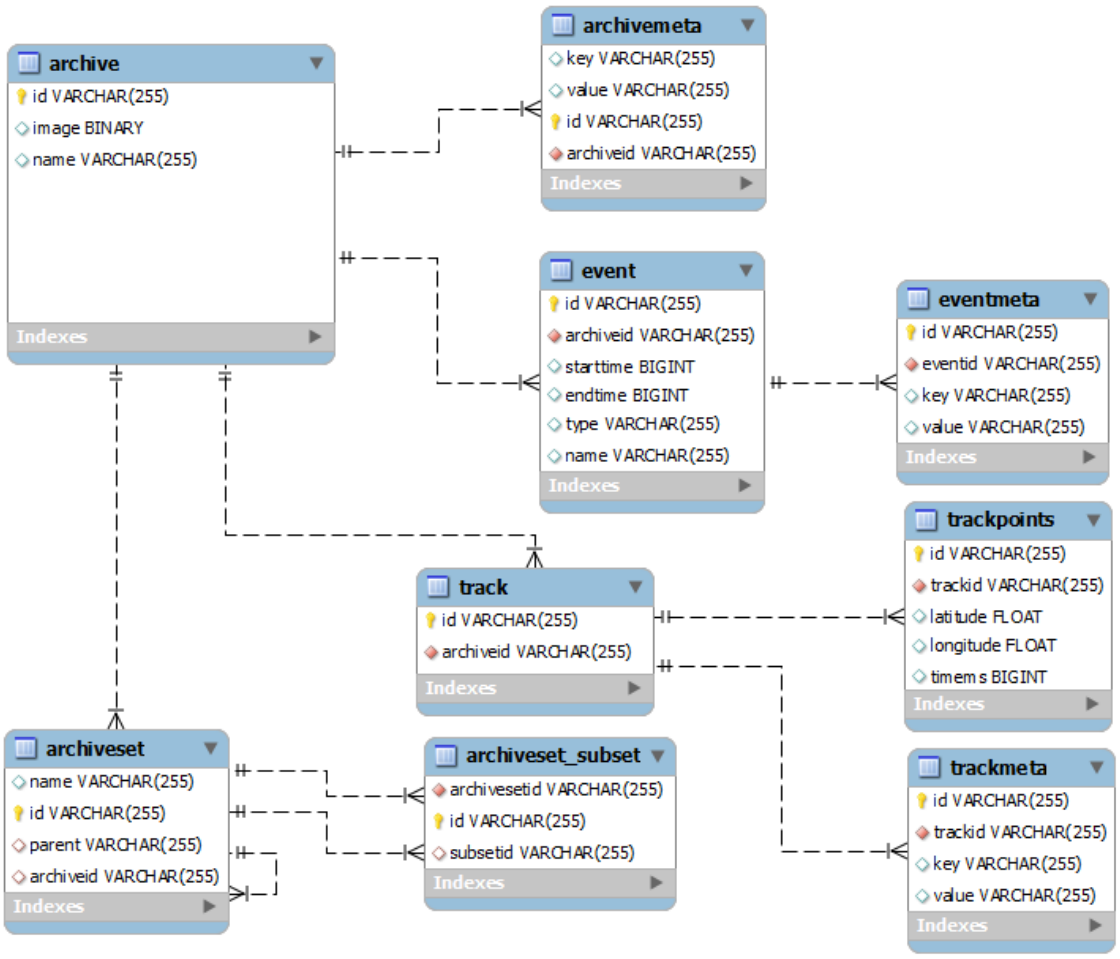


Figure 1: Database Structure for Archives and Analysis Sets

The database stores archives which represent vessels. Each archive can have a name, id, image, set of meta data and set of events.

2.6 Database Structure for Archives and Links

The database may also be used to store links between archives. Each link has a unique id, type, name, the two archive ids it connects, and possibly a weight. It also contains an analysis set id. This allows us to associate a set of links as part of the same analysis. See Figure 2.

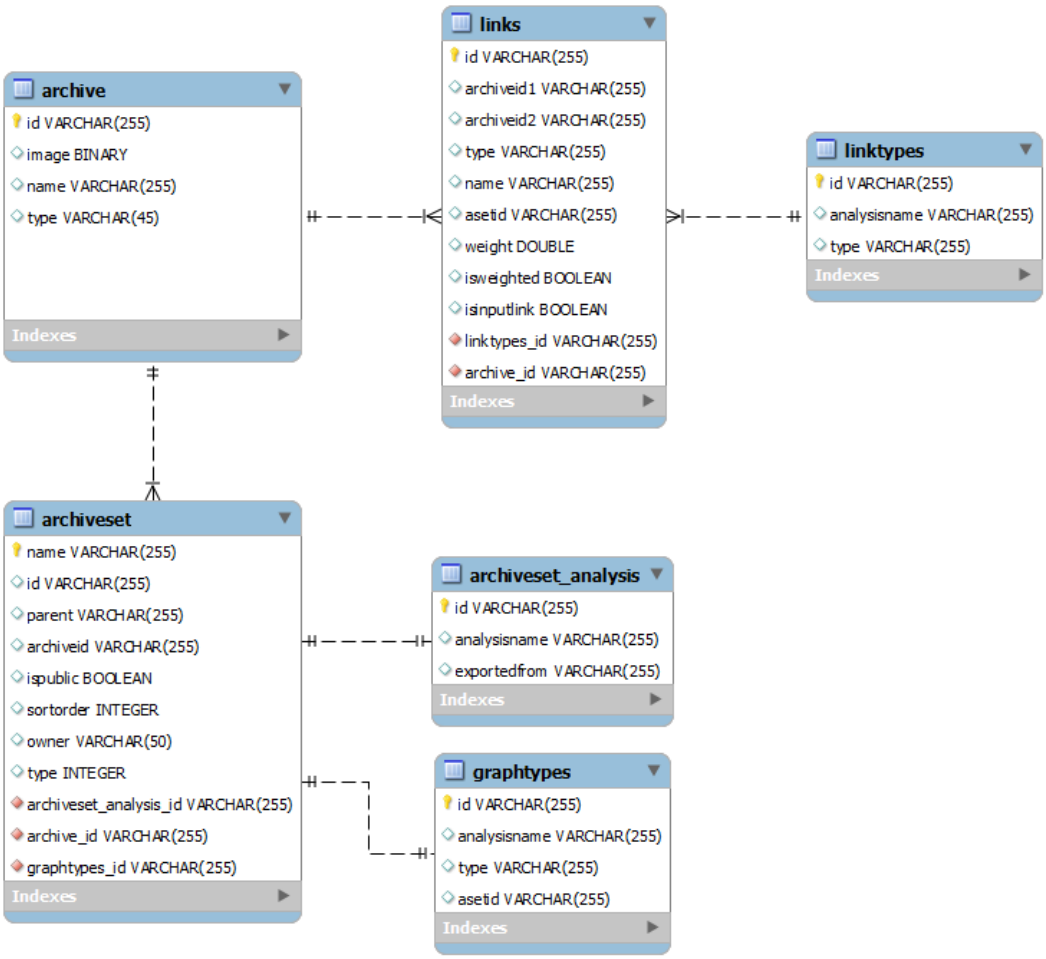


Figure 2: Database Structure for Archives Links

The database stores links which represent connections between the archives.

2.7 REST Data Objects

The following Java types are used by the REST functions. These types are automatically converted to JSON or XML as required at runtime using the same field names. Note that "ArrayList" objects are converted to JSON arrays and "HashMap" objects become JSON objects with a single property "entry" whose value is a list of JSON objects with "key" and "value" properties.

```
public class Archive {
	public String name;
	public String id;
}

public class ArchiveList {
	private ArrayList<Archive> archives;
}
public class ArchiveIDList {
```

```
	private String[] archiveids;
}
public class ArchiveData {
	private HashMap<String, String> properties;
}
public class ArchiveRestData {
	public byte[] image;
	public HashMap<String, String> metaData;
}
public class ArchiveEvent {
	public String id;
	public String archiveid;
	public String name;
	public String type;
	public long start;
	public long end;
	public HashMap<String,String> meta;
}
public class EventList {
	private ArrayList<ArchiveEvent> events;
}
public class AnalysisSetList {
	private ArrayList<AnalysisSet> sets;
}
public class AnalysisSet {
	public String setID;
	public String name;
	public ArrayList<AnalysisSet> children;
	public String archiveID;
	public String parentID;
	public int type;
	public int sortOrder;
	public Boolean isPublic;
}
public class TracksRestData {
	public String archiveid;
	public String name;
	public String type;
	public String archiveType;
	public TrackRestData[] tracks;
}
public class TrackRestData {
	public String archiveid;
	public PointRestData[] points;
}
public class PointRestData {
	public double x;
	public double y;
	public double t;
}
public class ArchiveCondition {
	private String field;
	private String compare;
	private String value;
}
```

```
public class EventCountRequest {
	private String[] archives;
	private ArchiveCondition[] conditions;
}
public class ArchiveDataAndEvents {
	private Archive archive;
	private ArrayList<ArchiveEvent> events;
	private HashMap<String, String> properties;
}
public class SetDataAndEvents {
	private ArrayList<ArchiveDataAndEvents> archivelist;
	private ArrayList<Link> links;
}
public class TimeRegionFilter {
	private String[] archives;
	private String left;
	private String right;
	private String top;
	private String bottom;
	private String start;
	private String end;
	private String[] uncheckedEventTypes;
}
public class StringPair {
	private String key;
	private String value;
}public class StringList {
	private String[] strings;
}
public class GPWRecord {
	String name;
	String flag;
	String category;
	String mmsi;
	String uid;
	String callSign;
	long posDate;
	double posLat;
	double posLon;
	double course;
	double speed;
	long msgDate;
	String imo;
}
public class GPWPosDate {
	public long posDate;
	public double posLat;
	public double posLon;
}
public class GPWVesselTracks {
	HashMap<String, ArrayList<GPWPosDate>> vesselMap;
}
public class GPWEncounter {
	private ArrayList<String> mmsis;
	private double latitude;
```

```
	private double longitude;
	private long time;
}
public class Link {
	private String id;
	private String archiveid1;
	private String archiveid2;
	private String type;
	private String name;
	private Double weight;
	private String asetid;
}
public class LinkList {
	private ArrayList<Link> links;
}
public class GraphImportRequest {
	String analysisName;
	String[] inputFileNames;
	String[] outputFileNames;
	String[] tableFileNames;
	String[] inputFileIds;
	String[] outputFileIds;
	String[] tableFileIds;
	boolean scrapeWiki;
	boolean reverseGeocode;
}
public class NodeLinkResult {
	private ArrayList<Link> links;
	private ArrayList<StringPair> nodes;
	private List<String> linkTypes;
	private Map<String,LinkList> nodeBasedLinks;
	private ArrayList<String> nodeBasedLinkTypes;
	private ArrayList<String> inputLinkTypes;
}
public class RelatedLinksRequest {
	private String[] archiveIds;
	private String setId;
	private String[] linkTypes;
}
public class TracksAndLinks {
	TracksRestData []tracks;
	NodeLinkResult graph;
}
public class TracksAndLinksRequest {
	String archiveId;
	String setId;
}
```

2.8 REST URLs

The following REST services are provided:

Operation: GET

URL	http://<was_url>/rest/archive/archives
Result Type	ArchiveList
Description	Returns a complete list of all archives in the system including names and ids.

URL	http://<was_url>/rest/archive/archivedata/{archiveid}
Result Type	ArchiveData
Description	Returns the meta data (key,value) pairs for the given archiveid

URL	http://<was_url>/rest/events/events/{archiveid}
Result Type	EventList
Description	Returns a list of all events for the given archiveid.

URL	http://<was_url>/rest/archive/analysissets
Result Type	AnalysisSetList
Description	Returns a tree structure of analysis sets.

URL	http://<was_url>/rest/archive/path/{archiveid}
Result Type	KML
Description	Returns a KML representation of the tracks for the given archive.

URL	http://<was_url>/rest/archive/encounters/{archiveid}.json
Result Type	TracksRestData
Description	Returns close encounter tracks for the given archive.

URL	http://<was_url>/rest/archive/encounters/{archiveid}/{start}/{end}.png
Result Type	image/png
Description	Returns an image of the close encounter tracks for the given archive and time range in milliseconds.

URL	http://<was_url>/rest/archive/vesselmap/{start}/{end}/{left}/{top}/{right}/{bottom}
Result Type	ArchiveList
Description	Returns a complete list of all archives in the system with tracks intersecting the given time and lat/lon bounds.

URL	http://<was_url>/rest/archive/setdata/{setid}
Result Type	SetDataAndEvents
Description	For every archive contained in the given set, returns all meta data and events.

URL	http://<was_url>/rest/archive/setkml/{setid}
Result Type	KML datafile
Description	Creates a kml file containing tracks for every archive contained in the given set.

URL	http://<was_url>/rest/archive/setcsv/{setid}
Result Type	CSV datafile
Description	Creates a csv file containing meta data for every archive contained in the given set.

URL	http://<was_url>/rest/archive/wordcloud/{archiveid}/{width}_{height}
Result Type	String URL
Description	Generates a word cloud from the ports visited by the given archiveid and return a url to the image.

URL	http://<was_url>/rest/archive/distinctarchivemeta
Result Type	StringList
Description	Returns a list of all archive meta data keys in the database.

URL	http://<was_url>/rest/archive/refreshautogeneratedset/{setid}
Result Type	N/A
Description	Launches the ISTIP service to update the autogenerated analysis set.

URL	http://<was_url>/rest/eventimage/{type}.png
Result Type	Image/png
Description	Fetches the image associated with the given event type.

URL	http://<was_url>/rest/event/eventtypes
Result Type	StringList
Description	All of the event types in the event type database.

URL	http://<was_url>/rest/archive/links/{setid}/{maxWeightedLinksPerNodeType}
Result Type	NodeLinkResult
Description	Fetches node-link data for the analysis set with id={setid} and returns {maxWeightedLinksPerNodeType} analytic links per similarity node, per analysis metric

URL	http://<was_url>/rest/archive/links/{setid}
Result Type	NodeLinkResult
Description	Fetches node-link data for the analysis set with id={setid} and returns 500 analytic links per similarity node, per analysis metric

URL	http://<was_url>/rest/graphimport/importedanalysisnames
Result Type	String
Description	Returns a list of all analysis names that have been imported into the database

Operation: POST

URL	http://<was_url>/rest/event/events
Posted Data	ArchiveIDList
Result Type	EventList
Description	Fetches events for all of the archives specified in the archive id list.

URL	http://<was_url>/rest/archive/tracks.json
Posted Data	ArchiveIDList
Result Type	TracksRestData[]

Description	Fetches tracks for all archives in ArchiveIDList.

URL	http://<was_url>/rest/archive/encounters.json
Posted Data	ArchiveIDList
Result Type	TracksRestData[]
Description	Fetches close encounter tracks for all archives in ArchiveIDList.

URL	http://<was_url>/rest/archive/vessels
Posted Data	Text file (tab delimited)
Result Type	None
Description	Reads a tab delimited text file containing new vessel data into the database. The first line of the file must contain headers including “VESSEL_NAME”.

URL	http://<was_url>/rest/archive/vesseldata
Posted Data	Text file (tab delimited)
Result Type	None
Description	Reads a tab delimited text file containing vessel meta data into the database. The first line of the file must contain headers including “ID” for the vessel archive id.

URL	http://<was_url>/rest/events/events
Posted Data	Text file (tab delimited)
Result Type	None
Description	Reads a tab delimited text file containing event data into the database. The first line of the file must contain headers. ARCHIVE_ID should be the first column. Name, Type and EVENT_DATE should be included amongst the headers.

URL	http://<was_url>/rest/archive/kmltracks/{prefix}/{setname}
Posted Data	Text file (KML)
Result Type	None
Description	Reads a KML datafile similar to those in the ISTIP DataInjector project. Vessels and tracks will be created accordingly.

URL	http://<was_url>/rest/archive/listdata
Posted Data	ArchiveIDList
Result Type	SetDataAndEvents
Description	For every archive in the given list, returns all meta data and events.

URL	http://<was_url>/rest/archive/listdata
Posted Data	TimeRegionFilter
Result Type	SetDataAndEvents
Description	For every archive in the given list, returns all meta data and events subject to the time/region filter.

URL	http://<was_url>/rest/archive/filteredtracks.json
Posted Data	TimeRegionFilter
Result Type	TracksRestData
Description	For every archive in the given list, returns all tracks subject to the time/region filter.

URL	http://<was_url>/rest/archive/addArchive/{name}/{id}/{archivetype}
Posted Data	StringPair[]
Result Type	N/A
Description	Creates an archive with the given name, id, type and posted meta data.

URL	http://<was_url>/rest/archive/meta/{key}
PostedData	ArchiveIDList
Result Type	None
Description	Clears the given key from the meta data for the all archives.

URL	http://<was_url>/rest/archive/settracks/{archiveid}
Posted Data	ArrayList<GPWRecord>
Result Type	None
Description	Creates a track for the given archive id with the given point list.

URL	http://<was_url>/rest/archive/scrape
Posted Data	ArchiveCondition[]
Result Type	ArchiveData
Description	Scrapes the Transport Canada website for meta data associated with the given archives. Archives should have mmsi, imo or name fields for the TC search.

URL	http://<was_url>/rest/eventimage/{type}
Posted Data	image/png
Result Type	N/A
Description	Sets the image for the given event type.

URL	http://<was_url>/rest/events/filteredevents
Posted Data	TimeRegionFilter
Result Type	EventList
Description	Fetches events for all of the archives specified in the filter within the time/region boundaries.

URL	http://<was_url>/rest/event/eventcounts
Posted Data	EventCountRequest
Result Type	ArchiveData
Description	Returns a hashmap of {archiveid,count} for the given event count request.

URL	http://<was_url>/rest/event/eventcounts/{start}/{end}
Posted Data	EventCountRequest
Result Type	ArchiveData
Description	Returns a hashmap of {archiveid,count} for the given event count request bounded by the given start/end times.

URL	http://<was_url>/rest/archive/relatedlinks
Result Type	NodeLinkResult
Description	Returns a NodeLinkResult for the specified RelatedLinksRequest object

URL	http://<was_url>/rest/archive/tracksandlinks
Result Type	TracksAndLinks
Description	Returns a TracksAndLinks object (TracksRestData + NodeLinkResult) for the specified archiveid and setid

URL	http://<was_url>/rest/archive/verifyaddanalysissets/{asetid}
Result Type	Boolean (as String)
Description	Returns true if archives in ArchiveIDList can be added to {asetid}, false if we are mixing archives from multiple analysis names

URL	http://<was_url>/rest/graphimport/fromfiles
Result Type	String
Description	Returns the first autogenerated asset id for input graph provided in GraphImportRequest. This will import a scenario (input graphs + output graphs + output tables) into the system and store it by an analysis name.

Operation: PUT

URL	http://<was_url>/rest/archive/analysisset/{name}/{parentid}
Result Type	String
Description	Creates a new analysis set with the given name and parent and returns the new id.

URL	http://<was_url>/rest/archive/archive/{name}/{id}
PostedData	ArchiveRestData
Result Type	None
Description	Creates a new archive with the given name, id, meta data and image.

URL	http://<was_url>/rest/archive/meta/{archiveid}
PostedData	ArchiveRestData
Result Type	None
Description	Clears then sets the meta data for the given archiveid.

URL	http://<was_url>/rest/archive/image/{archiveid}
PostedData	String (image url)
Result Type	None
Description	Sets the image for the given archive by reading from the given url.

URL	http://<was_url>/rest/archive/analysisset/ {name}/{parentid}/{archiveid}
Result Type	String
Description	Creates a new analysis set which references the given archive. Used to add the referenced archive to the analysis set specified by parentid.

URL	http://<was_url>/rest/event/events
Posted Data	ArchiveEvent
Result Type	String

Description	Adds the new event to the database and returns the id. All event fields must be included in the posted data (which can be JSON or XML).

URL	http://<was_url>/rest/archive/allvessels
Result Type	None
Description	Creates a new analysis set entitled “All Vessels” containing all the vessels.

URL	http://<was_url>/rest/archive/queryanalysisset/{name}
Result Type	String
Description	Returns the id of an analysis set with the given name.

URL	http://<was_url>/rest/archive/analysisset/move/{id}/{parentid}/{newparentid}/{index}
Result Type	N/A
Description	Moves the analysis set with the given id for parented to newparentid. Order amongst the children of the new parent is determined by index.

URL	http://<was_url>/rest/archive/analysisset/rename/{id}/{name}
Result Type	N/A
Description	Renames the analysis set with the given id.

URL	http://<was_url>/rest/archive/autogeneratedaset/{name}/{parentid}/{archiveid}
Result Type	String
Description	Creates an autogenerated analysis set with the given name, parent and referenced archiveid. Returns the id of the new analysis set.

URL	http://<was_url>/rest/archive/autogeneratedaset/{name}/{parentid}
Result Type	String
Description	Creates an autogenerated analysis set with the given name and parent. Return the id of the new analysis set.

URL	http://<was_url>/rest/archive/autogeneratedaset/{name}
Result Type	String
Description	Creates an autogenerated analysis set with the given name. Returns the id of the new analysis set.

URL	http://<was_url>/rest/archive/meta/{archiveid}/{key}
Result Type	None
Description	Clears the given key from the meta data for the given archiveid.

URL	http://<was_url>/rest/archive/meta/{archiveid}/{key}/{value}
Result Type	None
Description	Sets the given key/value for the meta data for the given archiveid.

URL	http://<was_url>/rest/archive/associateanalysisset/{newasetid}/{originalasetid}
Result Type	None
Description	Gives the analysis set specified by {newasetid} the same associate as the one specified by {originalasetid}. This is used when we export a subset of another

	analysis set and we wish to preserve any links that exist between the archives.

Operation: DELETE

URL	http://<was_url>/rest/archive/analysisset/{id}
Result Type	none
Description	Deletes the analysis set with the given id.

URL	http://<was_url>/rest/archive/archive/{id}
Result Type	None
Description	Deletes the archive with the given id.

URL	http://<was_url>/rest/archive/archives
Result Type	None
Description	Deletes all archives

URL	http://<was_url>/rest/event/events/{id}
Result Type	None
Description	Deletes the event with the given id.

URL	http://<was_url>/rest/event/events
Result Type	None
Description	Deletes all events.

URL	http://<was_url>/rest/archive/tracks
Result Type	None
Description	Deletes all tracks

URL	http://<was_url>/rest/archive/removeautogenchildren/{id}
Result Type	None
Description	Deletes all tracks

URL	http://<was_url>/rest/archive/deleteasetchildren/{id}
Result Type	None
Description	Deletes the children from the analysis set with the given id.

URL	http://<was_url>/rest/eventimage/delete/{type}
Result Type	None
Description	Removes the given event type from the event image database.

URL	http://<was_url>/rest/archive/killall
Result Type	None
Description	Clears the entire database, including all archives, analysis sets, links, tracks, etc.

2.9 GPW REST calls

Operation: GET

URL	http://<was_url>/rest/gpw/loiterers/{start}/{end}/{left}/{top}/{right}/{bottom}
Result Type	ArrayList<GPWRecord>
Description	Fetches all of the entries in the loiterers table within the time/region bounds.

URL	http://<was_url>/rest/gpw/encounters/{start}/{end}/{left}/{top}/{right}/{bottom}
Result Type	ArrayList<GPWRecord>
Description	Fetches all of the entries in the encounters table within the time/region bounds.

URL	http://<was_url>/rest/gpw/mmsiencounters/{mmsi}
Result Type	ArrayList<GPWRecord>
Description	Fetches all of the entries in the encounters table with the given mmsi.

URL	http://<was_url>/rest/gpw/gpwtrackdata/{mmsi}/{start}/{end}
Result Type	String
Description	Fetches a text description of the track data for the given mmsi and time range.

URL	http://<was_url>/rest/gpw/latest/{start}/{end}/{left}/{top}/{right}/{bottom}
Result Type	ArrayList<GPWRecord>
Description	Gets the latest position of all vessels reporting within the given time/region bounds.

URL	http://<was_url>/rest/gpw/gpwtrack/{field}/{value}/{start}/{end}/{left}/{top}/{right}/{bottom}
Result Type	ArrayList<GPWRecord>
Description	Gets the track for a vessel matching (field==value) bounded by the given time/region.

URL	http://<was_url>/rest/gpw/gpwtrack/{field}/{value}/{start}/{end}
Result Type	ArrayList<GPWRecord>
Description	Gets the track for a vessel matching (field==value) bounded by the given times.

URL	http://<was_url>/rest/gpw/gpwfilteredtrack/{field}/{value}/{start}/{end}
Result Type	ArrayList<GPWRecord>
Description	Gets the track for a vessel matching (field==value) bounded by the given times. Removes collinear points to reduce data volume.

Operation: POST

URL	http://<was_url>/rest/gpw/csv
Posted Data	CSV File
Result Type	N/A
Description	Reads the provided CSV file of GPW records into the gpwdata database table.

Operation: DELETE

URL	http://<was_url>/rest/gpw/all
Result Type	None
Description	Clears the gpwdata table.

2.10 GWT and JavaScript Integration

Oculus initially developed MVAP using GWT and the WAS framework for the high-level objects such as windows and client-server communication. The UI components for the widgets within the windows were developed using JavaScript, jQuery and the Aperture library. In Phase 3, the system was rearchitected so that it was possible to deploy the entire system without using GWT or WAS.

The MVAP project provided an excellent opportunity to compare development directly in JavaScript with GWT development. For Java developers learning web development, GWT development seems easier since coding can be performed in Java which is a familiar environment with all of the useful features of a structured programming language. After implementing some significant user interface components directly in JavaScript and becoming more familiar with the environment, the team began to prefer this style of development. Eliminating the compile time required by GWT allows for rapid development cycles and the scripting language along with tools such as jQuery allow for rapid UI development with minimal code. The end result was that coding directly in JavaScript produced better results in terms of performance, aesthetics and reduced development time.

2.11 jQuery JavaScript Library

The JavaScript code in the MVAP project makes extensive use of the jQuery library. This web standard library provides useful tools for creating consistent user interface components and for implementing interaction features such as drag and drop with a remarkably small amount of coding effort.

2.12 Aperture JavaScript Library

The Aperture library is developed by Oculus as an open source visualization framework available under the MIT Open Source License. MVAP makes use of Aperture to provide the scatter plot behind Magnets Grid and to create the event timelines. In both cases, the MVAP source code only needs to update the JSON data to be visualized by these components and then call an update method to update the visual. Aperture handles rendering and interaction with the data.

The Aperture library could be used by other developers building on the WAS framework to rapidly produce high performance, efficient information display, and aesthetically pleasing data visualizations such as charts and graphs.

2.13 Generic Widget Architecture

We have provided four generic widgets created in JavaScript and extended in the WAS Framework for use in MVAP: Magnets Grid, Visual Summary Cards, Timeline, and Map and Timeline. Our generic widgets all operate under a common style. Data is provided to the widgets using JSON while we provide event callbacks and methods to extract data from the widgets. The source code

for the generic widgets must be included in the welcome file (i.e. wasPrototype.jsp) and is contained in the files:

```
aperture-dustandmagnets.js
aperture-dustandmagnets-customAttributes.js
aperture-dustandmagnets-marquee.js
aperture-dustandmagnets-tabs.js
aperture-summarycard.js
aperture-timeline.js
aperture-multitimeline.js
aperture-mapandtimeline.js
aperture-mapcontrol.js
aperture-maplayercontrol.js
aperture-maplayerdialog.js
```

Since our generic widgets make use of jQuery and jQueryUI, these must also be considered dependencies.

2.13.1 Visual Summary Card

The JSON schema for creating a Visual Summary card is shown below:

```
{
	"name" : "VSC",
	"type" : "array",
	"items":
	[
		{
			"$ref" : "header"
		},
		{
			"$ref" : "image",
			"optional" : true
		},
		{
			"$ref" : "pair",
			"optional" : true
		},
		{
			"$ref" : "text",
			"optional" : true
		},
		{
			"$ref" : "url",
			"optional" : true
		},
		{
			"$ref" : "graph",
			"optional" : true
		},
		{
			"$ref" : "multigraph",
			"optional" : true
		},
		{
```

```
                    "$ref" : "map",
                    "optional" : true
            },

      ]
}
```

This defines an array of visual elements that are to be laid out on the card. The header is a unique element that provides any common data across both sides of visual summary cards (i.e./ a title). The header is defined as follows:

Header

```
{
      "name" : "header",
      "type" : "object",
      "properties" :
      {
            "type": "header",
            "title" : "string",
            "flip" : "string",
      }
}
```

Our visual elements are all defined similarly, with each specifying their own unique properties. All visual elements contain an (x,y) position relative to the top left corner of the card as well as a width and height. Additionally, they each contain a value side which specifies whether the element appears on the front of the card (true) or the back of the card (false). The JSON schemas for creating our four types of visual elements are as follows:

Image

```
{
      "name" : "image",
      "type" : "object",
      "properties" :
      {
            "type": 'image',
            "uri" : "string",
            "x" : "number",
            "y" : "number"
            "width" : "number",
            "height" : "number",
            "side" : "Boolean"
            "style" : "string"
            "title" : "string",
      }
}
```

Link

```
{
      "name" : "link",
      "type" : "object",
      "description" : "An anchor HTML element"
```

```
      "properties":
      {
            "type" : 'text',
            "url": "string",
            "x" : "number",
            "y" : "number"
            "width" : "number",
            "height" : "number",
            "side" : "Boolean"
      }
}
```

Pair

```
{
      "name" : "pair",
      "type" : "object",
      "description" : "A pair of text elements side by side. Used as
      table elements"
      "properties":
      {
            "type" : 'pair',
            "label": "string",
            "value": "string",
            "labelWidth": "number",
            "x" : "number",
            "y" : "number"
            "width" : "number",
            "height" : "number",
            "side" : "Boolean"
      }
}
```

Text

```
{
      "name" : "text",
      "type" : "object",
      "description" : "A simple text element"
      "properties":
      {
            "type" : 'text',
            "full_text": "string",
            "x" : "number",
            "y" : "number"
            "width" : "number",
            "height" : "number",
            "side" : "Boolean",
      }
}
```

Graph

```
{
      "name" : "graph",
      "type" : "object",
      "description" : "A node-link graph element"
      "properties":
```

```
        {
        type : "graph",
        x : left,
        y : top,
        archiveid: archiveid,
        width : width,
        height : height,
        graph : graph,
        side : side
        }
}
```

Multigraph

```
{
        "name" : "multigraph",
        "type" : "object",
        "description" : "A element that contains multiple graphs"
        "properties":
        {
            type : "multigraph",
            x : left,
            y : top,
            archiveid: archiveid,
            width : width,
            height : height,
            graphs : graphs,
            types : types,
            side : side
        }
}
```

Map

```
{
        "name" : "map",
        "type" : "object",
        "description" : "A map element"
        "properties":
        {
              type: "map",
              width: width,
              height: height,
              x: left,
              y: top,
              lat: lat,
              lon: lon,
              uri: URL,
              iconSize: iconSize,
              side: side,
              trackArray: trackArray
        }
}
```

In order to instantiate an Aperture Summary Card, we must provide the id of the div element that will be drawn in (known as the canvas) and a unique identifier. Ex:

```
var card = new aperture.summarycard.SummaryCard({id: "#canvas", uuid:
1234});
```

We can then set the JSON data on our card object and call update() to render our summary card. In order to provide functionality in external environments, we have set up event callbacks that may be passed into our Visual Summary Card to notify us of any events.

```
setFlipCallback : function(callback)
setSelectCallback : function(callback)
setDragStopCallback: function(callback)
```

These functions take callback functions as arguments. The callbacks will be called when we flip a card, select a card, or drag a card. We used these event callbacks to integrate the Visual Summary Card widget into our Record Browser GWT widget. The flip and scroll events were used to synchronize the sides and scroll positions of our two summary cards. The select event was used to keep track of which cards were selected at any given time. The drag stop event was used to implement our drag and drop functionality for comparing cards.

2.13.2 Magnets Grid

Unlike the Visual Summary Card JSON data, the data required for Magnets Grid must be split up into multiple components. There are three main pieces of data required to correctly instantiate a Magnets Grid widget:

- Dust data
- Magnet data
- Attribute data

We will start with our dust data. This contains all elements of the data set we're looking to extract information about. Each dust item is an element of an array that defines our entire data. Within the dust item, we must specify a name and id for the item, as well as an array of properties. These properties are key/value pairs where key is simply a string, and value is any type of JavaScript object (string, number, Boolean, etc). Formally, the JSON schema is defined as follows:

```
{
      "name" : "dustData",
      "type" : "array",
      "description" : "An array of dust items, our most basic form of
data."
      "items" :
      [
            {
                  "name" : "dust",
                  "type" : "object",
                  "properties" :
                  {
```

```
                        "name" : "string",
                        "id" : "string",
                        "properties" :
                        {
                              "type" : "object",
                              "description" : "Arbitrary data attributes"
                        }
                  }
            }
      ]
}
```

Magnet data and attribute data define the semantics of how the widget operates. That is, these two pieces of data tell the widget what type of data is stored in each dust item, and how it should be treated. Magnet data is used to create draggable and droppable magnets in the plot area that are used to separate the dust particles by forces of attraction. We define magnet data as follows:

```
{
      "name" : "magnetData",
      "type" : "array",
      "items" :
      [
            {
                  "name" : "magnet",
                  "type" : "object",
                  "properties" :
                  {
                        "displayName" : "string",
                        "type":"enum":["Numerical","Boolean"],
                        "attributeAccessName": "string"
                  }
            }
      ]
}
```

Attribute access name corresponds to a key in the property bag defined in the dust items. The type of the magnet can be either numerical or boolean. Numerical magnets are assumed to attract dust with larger values faster than dust with smaller values. Boolean magnets operate on a true/false basis and will attract only dust items with the desired attribute as true.

In addition to magnets, we can also specify a broader range of attributes. We use this general attribute data to separate dust by axis. These also allow us to add a color/size filter to the dust at any given time. We specify an array of categorical and numerical attributes. Formally:

```
{
      "name" : "attributes",
      "type" : "array",
      "items" :
      [
            {
                  "$ref" : "categoricalAttribute"
            },
```

```
            {
                  "$ref" : "numericalAttribute"
            }
      ]
}
```

In Phase 3, Magnets Grid was extended to permit being launched with no data. The visualization can then be populated by dragging a csv file into the interface. Each row in the data becomes a dust element and the columns are used to create magnets and attributes.

2.13.3 Timeline

The JSON schema for creating a Timeline Widget is shown below. It is represented as an array with each element providing the data for one timeline item. Within the item, we must specify a name and id for the item. These properties are key/value pairs where key is simply a string, and value is any type of JavaScript object (string, number, Boolean, etc.). Formally, the JSON schema is defined as follows:

```
{
      "type": "array"
      "description": "Array with each element providing info for a
      timeline"
      {
            "type": "object"
            "name": "string"
            "id": "string"

            "tlCenter": "number"
            "tlZoom": "number"
            "tlStart": "number"
            "tlEnd": "number

            "events":
            [
                  {
                        "type": "string"
                        "end": "number"
                        "start": "number"
                        "iconUrl": "string"
                        "name": "string" <event name>
                        "properties":
                        [
                              {
                                    "key": "string"
                                    "value": "string"
                              }
                        ]
                  }
            ]
      }
}
```

As with the Visual Summary Card and Magnets Grid Widgets described above, we have provided an interface for extracting data from the Timeline Widget. We have the following event callbacks:

```
setSelectCallback : function(callback)
setLaunchFn : function(callback)
```

The select and launch callbacks both operate in a similar manner to their corresponding Magnets Grid callbacks. The select callback is called when the selection state of any timeline item in the widget changes. It expects an integer argument that represents the number of timelines selected. The launch function is called when we double click on a timeline item. It expects a function with a single argument, which is the id of the item we've double clicked. As in Magnets Grid, we use this to launch a Visual Summary Card from within the Timeline.

The Timeline also provides the following utility methods:

```
getTLData : function()
getSelectedIDs: function()
updateSize: function(w,h)
```

`getSelectedIDs` returns a JavaScript array of IDs of any selected Timeline items. `updateSize` is a utility function for scaling and resizing the Timeline window. If it is contained within a windowing system (such as WAS), update size will need to be called each time the containing window is resized. `getTLData` exports the timeline view state so it can be saved and restored by the workspace.

We can create a new Timeline with the following sample call:

```
var timeline = new
aperture.multitimeline.MultiTimeline("timelineContainer", buttonElem,
data, "tluuid1234");
```

This creates a Timeline widget in the div with the id "`timelineContainer`" and UUID "`tluuid1234`". Here, our JSON data is represented by the variables `data` and buttonElem represents a common button canvas where the Aperture Timeline can render its generic buttons.

2.13.4 Map and Timeline

The JSON schema for creating a Map and Timeline Widget includes the Timeline data schema as above along with a Map schema. Formally, the JSON schema for the Map is defined as follows:

```
{
      "type": "object",
      "properties": {
            "tracks": {
                  "type": "array",
                  "description": "Array of vessel ids",
                  "items": {
                        "type": "string"
                  }
            }
            "kmlzones": {
                  "type": "array",
```

```
                    "description": "Array of kml zones",
                    "items": {
                          "type": "object",
                          "properties": {
                                "name": "string",
                                "url": "string"
                          }
                    }
              }
        }
}
```

Map and Timeline has the following event callbacks which can be overridden:

```
tlIconClickFn : function(event) // called when a timeline icon is clicked
mapIconClickFn : function(event) // called when a map icon is clicked
mapDblIconClickFn : function(event) // called when a map icon is dbl
clicked
exportCallback : function(archiveids, allids) // called on export
```

The Map and Timeline also provides the following utility methods:

```
destroy: function()
resize: function(w,h)
```

`resize` is a utility function for scaling and resizing the window. If it is contained within a windowing system (such as WAS), update size will need to be called each time the containing window is resized. `destroy` should be called when the window is closed to clean up resources.

We can create a new Map and Timeline with the following call:

```
var timeline = new aperture.mapandtimeline.MapAndTimeline(element, id,
timelinedata, mapdata);
```

This creates a Map and Timeline widget in the given div "element" and UUID "`id`". Our JSON data for the timeline and map are separated in the variables timelinedata and mapdata.

In Phase 3, the Map and Timeline was extended so that it could be launched without any data specified. The user can then zoom to a time/region of interest and fetch vessels that have tracks within that region.

2.14 GPW Data Analysis

In Phase 3, Oculus received a set of GPW data including over 75,000 vessels and 112M position reports. Analysis of this data directly within the existing system proved to be unwieldy so new database tables, services and widgets were developed. The new widgets were tied into the existing tools via an export utility. Thus, a subset of the large dataset could be analysed using the existing tools.

2.14.1 Database structure

GPW data was provided as two separate tables. The track table contains one row for each track and the position table contains one row for each position report. There are many more columns than indicated in the diagram below.

In order to create an interactive visualization of loitering and close encounters, it was necessary to preprocess the data to create a table of loiterers and a table of close encounters. The loiterers table indicates the vessel mmsi and the time period during which it reported a fixed location. The encounters table uses an id field so that multiple entries represent a single encounter. There is one entry for each vessel involved.

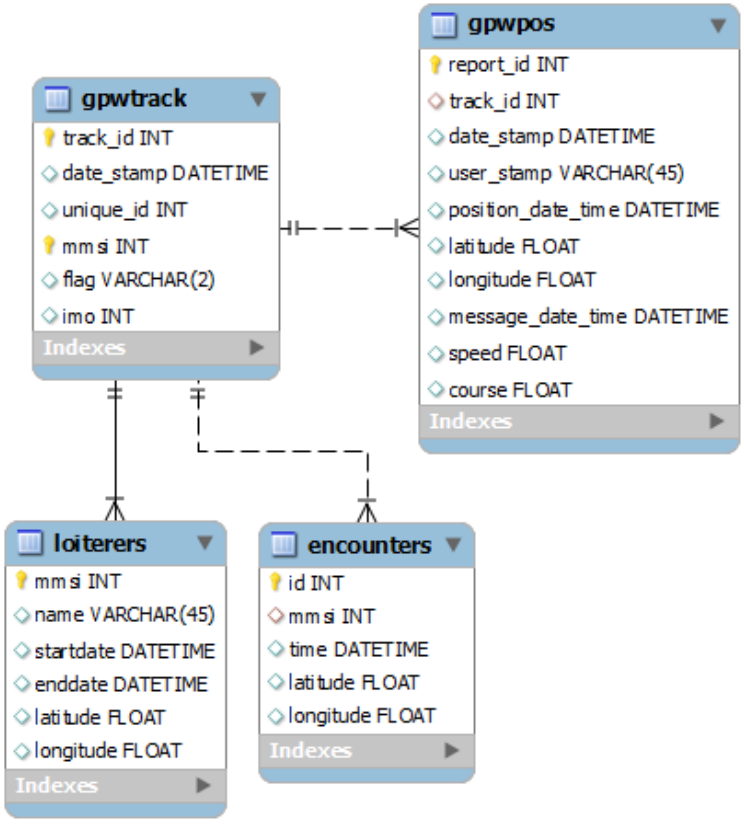


Figure 3: Database Structure for GPW Data Analysis
The track table contains one row for each track and the position table contains one row for each position report. Loiterers and encounters data is preprocessed for interactive visualization.

2.14.2 GPW preprocessing

The GPW data was provided as a set of gziped csv files. The project MVAP_DB contains java utilities for processing this data. The java utilities are just classes with main methods that when executed will read files and update database tables. They are intended to be used by technical staff, not the average user.

First, oculus.mvap.db.sources.gpw.GPWGZReader.java can be used to create the gpwtrack and gpwpos tables, unzip the csv files, read them into the tables and generate indices. This process can take a few hours depending upon the hardware used.

The class GPWDBReader was created to read an older format of GPW data into the gpwdata table. This approach is essentially deprecated.

The class GPWLoiteringAnalyzer can be used to read from gpwtrack and gpwpos and iterate over the positions of each vessel. When a vessel has multiple reports within the same region for an extended period, an entry is added to the loiterers table with the position, duration and mmsi.

The class GPWEncountersAnalyzer interpolates vessel positions at the end of each hour and if vessels are within the same grid square at that time, an entry is added to the encounters table.

2.14.3 GPW Map and Timeline Widget

The GPW Map and Timeline widget is similar to the MVAP Map and Timeline widget but accesses data from the gpwtrack, gpwpos, loiterers and encounters tables. The page is defined in the MVAP_UI project as war/mvap/GPWMapTimeline.html. The HTML makes use of the mvap-gpwmap.js JavaScript widget which in turn uses the aperture-gpwmapandtimeline.js script to instantiate the map and provide interaction with the gpw tables.

3 Design of the Apps

3.1 MVAP Functionality

This section describes the main functionality of widgets included in the MVAP. The widgets can be instantiated within the WAS framework or deployed as independent web pages. As independent web pages they can be used within frameworks such as SitScape or OWF.

Initial application of the widgets was for maritime domain awareness, in which archives represent vessels. For the SNA amendment, in addition to vessels, archives can also represent people or organizations. However, the original MVAP widgets were designed to be generic, and figures are included below to show use of the widgets for both maritime and social network analytics. The only functional change required to support SNA was an adaptation of the Vessel Summary Card to better display archives of the person type. This is described in the section on Person Summary Card.

The sections below describe the functionality of each widget.

3.1.1 Analysis Set Manager

The Analysis Set Manager opens when the application first opens. It allows users to view, update, create, and delete sets of archives (Analysis Sets) that can then be examined in other widgets.

Expand a set to display all archives in that group using the ⊞ button. To hide archives, click the ⊟ button.

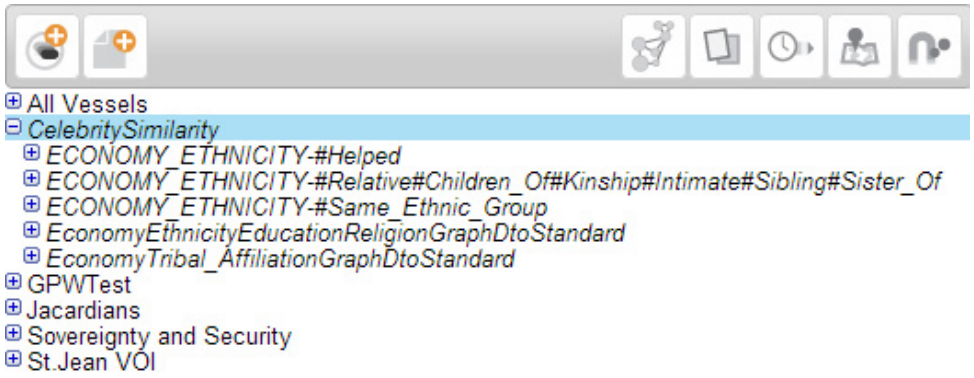


Figure 4: Analysis Set Manager
The Analysis Set Manager opens when MVAP starts up. It allows users to view, add, and modify groups of archives. It is also the launching pad for other widgets.

3.1.1.1 Add, Update, Remove

Two buttons facilitate main functionality, from left to right these are: Add Archive Group, and Add Archive (to the selected Archive Group).

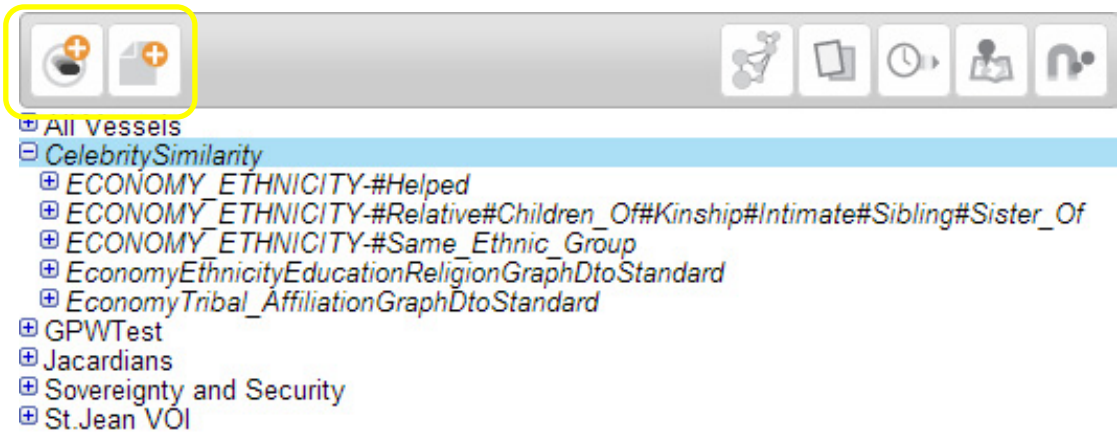


Figure 5: Analysis Set Manager: Add Ship Group & Add Ship buttons
Two buttons found in the top-left corner of the Analysis Set Manager (and elsewhere) allow the user to create new empty groups of archives, and to add archives to existing Archive Groups.

The add archive button will bring up the select archives dialog.

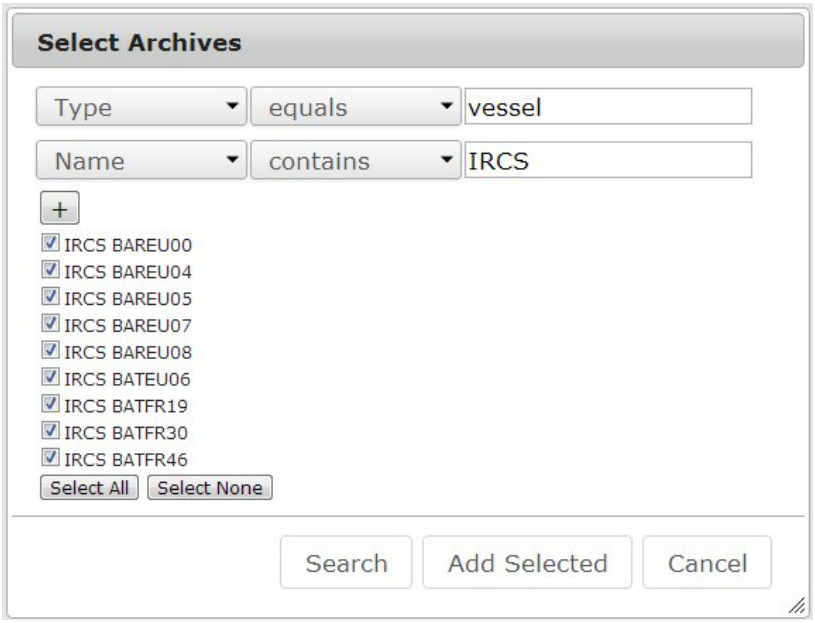


Figure 6: Select Archives Dialog

Multiple filters can be defined using the "+" button in the dialog. Archives that match the search criteria will be displayed after the "Search" button is pressed. Clicking on "Add Selected" will add all of the archives that are checked. Checks can be toggled with "Select All", "Select None" or individually by clicking on checkboxes.

An Archive or an Archive Group can be deleted when it is selected using the 'X' button to its right in the list.

3.1.1.2 Right click menu

The right click menu for analysis sets allows the user to rename/cut/copy/paste/delete items similar to the way a standard file system works. The export to KML option exports the tracks of the contained vessels in a format which can be read by Google Earth. Export CSV outputs the vessel data to a csv file which can be used by many third party tools (or Magnets Grid).

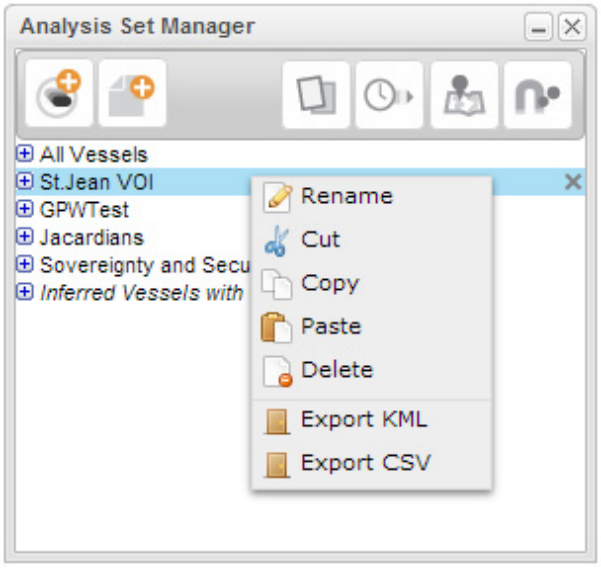


Figure 7: Analysis Set Manager Context Menu

3.1.1.3 ISTIP Artificial Reasoning Sets and Social Network Analysis created Sets

There are special sets created using ISTIP artificial reasoning and by importing GraphDTO files. Right clicking on a special set will trigger the reasoner and update the contents of the set. Both ISTIP sets and Social Network Analysis (SNA) created sets are read-only.

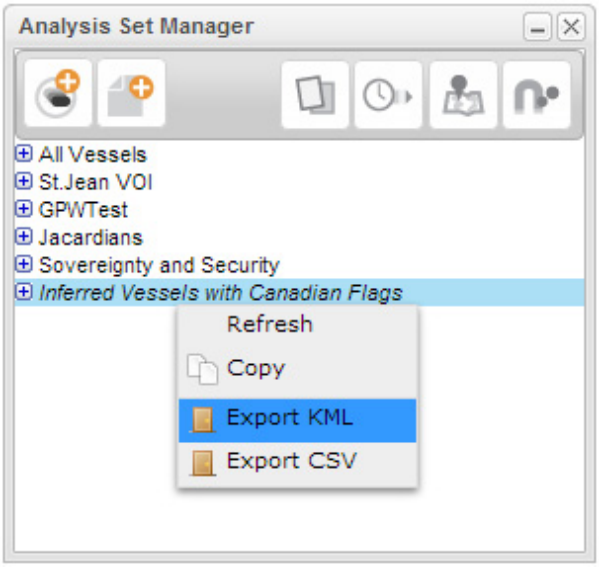


Figure 8: Autogenerated Set with Artificial Reasoning

3.1.1.4 Other Functions

The Analysis Set Manager also displays buttons to access the prototype's other main widgets with the selected analysis set:

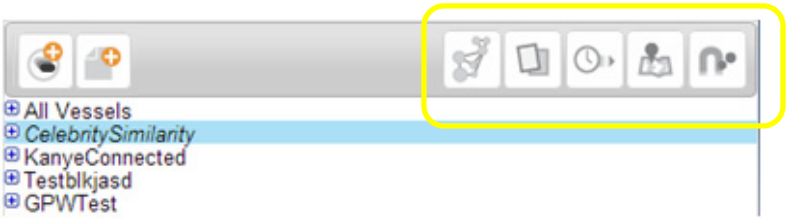


Figure 9: Analysis Set Manager: Widget Access
From the Analysis Set Manager, a user can open the Graph Analyzer, Record Browser, Timeline, Map, Magnets Grid for a selected Ship Group using the buttons in the top-right corner.

These widgets, from left to right are: **Graph Analyzer, Record Browser**, **Timeline**, **Map**, and **Magnets Grid**. More about these widgets are described in the selections below. Note that in order to open these widgets, a Ship Group (or else a group of other components like persons) must be selected (rather than a ship or a single person).

3.1.2 Vessel Summary Card

Double-clicking on any single vessel from the Analysis Set Manager opens up the Vessel Summary Card.

Figure 10: Vessel Summary Card

The Vessel Summary Card provides unique visual representation of each ship, suitable for rapid visual browsing and data drill-down*. A Vessel Summary Card can be accessed directly from the Analysis Set Manager by double-clicking on a ship item.*

The front side of this card shows a visual display of important ship information, including the vessel name, icons identifying vessel type, flag, and whether it is expected into a Canadian port (24hr or 96hr call-ins). The vessel image acts an immediate visual identifier. A Word Cloud below the vessel image displays place names the ship has connections with. Connections include ports visited, place of registry, owner address, etc. The larger the place name, the more connections a ship will have to that place. The Close Encounter Icon at the bottom left shows relative tracks of vessels that have been in close proximity. The Map Thumbnail at the bottom right shows the most recent location and tracks for the vessel.

The flip side of the card displays detailed information about that vessel in list format. Navigate back and forth between these sides using the 'FLIP' button in the bottom-right hand corner. Use the vertical scrollbar to view all the details.

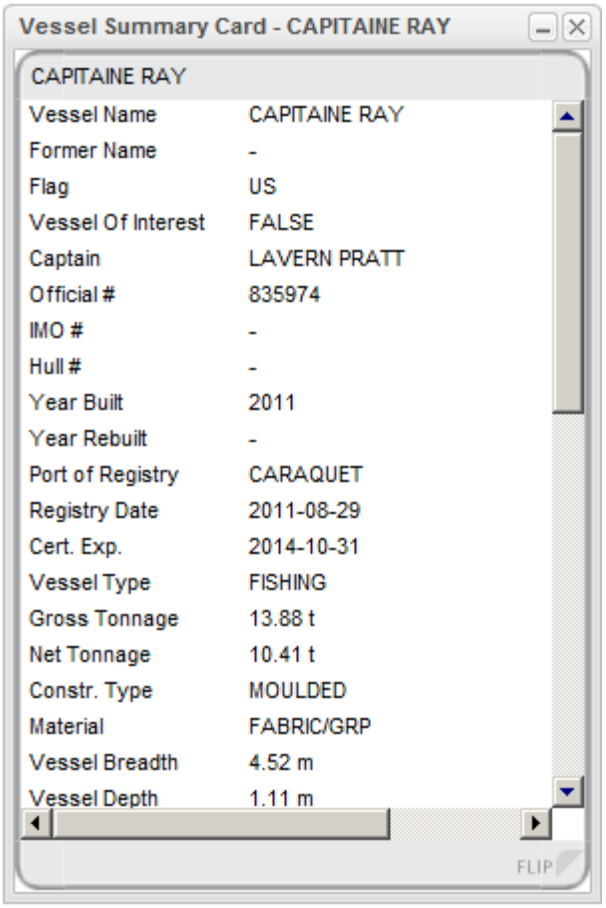


Figure 11: Vessel Summary Card: Flip Side
The "Flip" button on the bottom right of the Vessel Summary Card allows the user to access detailed information about the ship. Use the same button to return to the front side of the card.

The Vessel Summary Card can be accessed directly from the Analysis Set Manager (double-click a ship name), from within the Record Browser, from the Timeline (double-click a ship timeline or name), and from the Magnets Grid (double-click a ship/dust item).

3.1.3 Person Summary Card

The Person Summary Card is derived from the generic Aperture Summary Card specifically for display archives of type 'person'. Like the Vessel Summary Card, the flip side displays any information (in the form of meta data pairs) associated with that person. The front side of the card is slightly different as seen in the figure below:

Figure 12: Person Summary Card
The Person Summary Card provides unique visual representation of each person, suitable for rapid visual browsing and data drill-down. *A Person Summary Card can be accessed directly from the Analysis Set Manager by double-clicking on an archive item.*

The front side of this card shows a visual display of important person information, including the name. A Word Cloud below the person image displays any text from imported events associated with that person and any locations where the person has known to have been. The Graph Inset at the bottom left of the card shows any links this person has to other archives in the system. We can change the link type by using the drop down menu provided. The Map Inset shows any track this person and is centered on the last known whereabouts of the person.

3.1.4 Record Browser

Double-clicking on any Archive Group in the Analysis Set Manager brings up that group in the Record Browser. View summary cards for all archives in the group, navigating using the 'Next' and 'Previous' buttons.

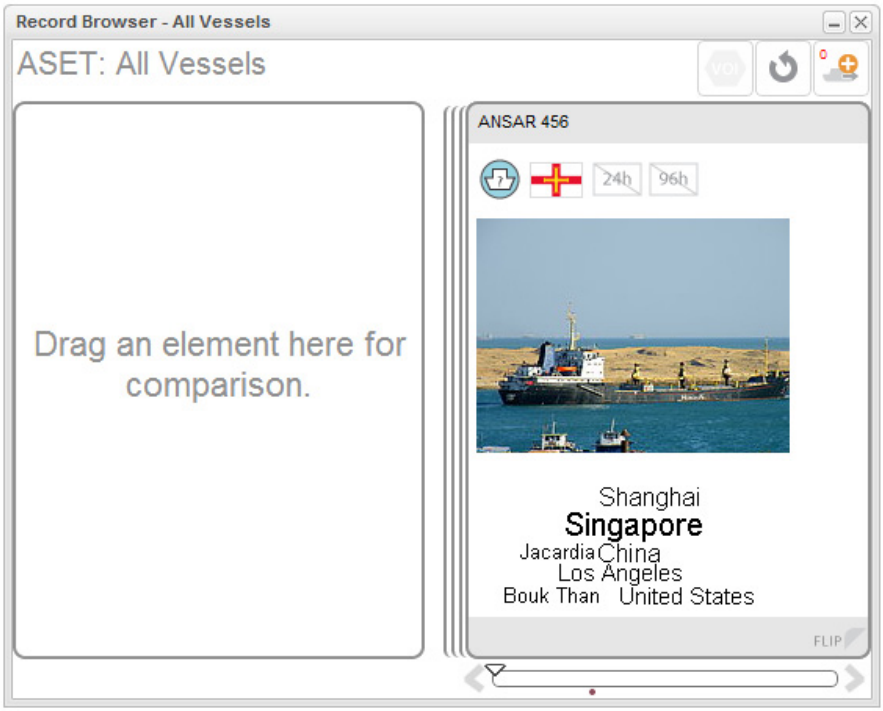


Figure 13: Record Browser

The Record Browser allows the user to quickly view ships in a group. Users can choose a comparison ship, select ships for export, and access the flip side of each vessel's summary card.

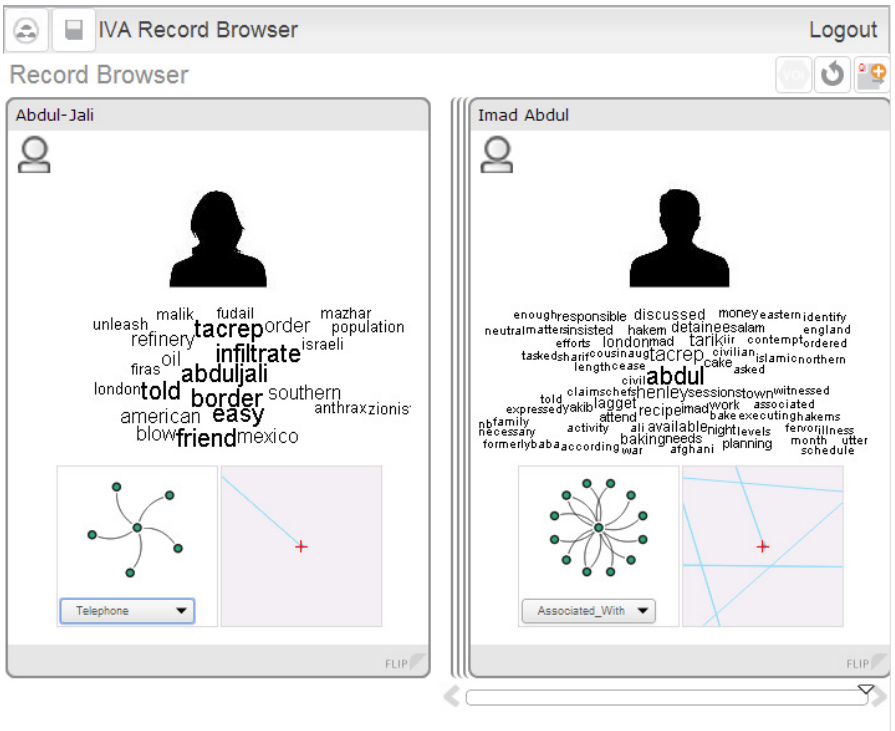


Figure 14: Record Browser for Social Network Analysis

When it is first opened, the Record Browser shows a single card on its right-hand side. The card on the right represents the 'Current' archive in the group. Navigate through the whole set using the 'Next', and 'Previous' buttons. Progress through the list is marked with the progress marker on the progress bar.

3.1.4.1 Comparison Card

The card on the left is used to 'hold' a ship for comparison with other ships (or a person against other persons). Create a Comparison Archive by dragging a Current Archive (click down on the grey bar near the archive's title) into the comparison area. This archive will remain the comparison archive until it is replaced, or until the browser is closed.

As with the Summary Card, further details are available on the reverse side of the card. The 'FLIP' button controls both the current and comparison cards.

3.1.4.2 Export Selected Archives

Any subset of archives can be exported from the current set and either added to an existing set, or a new set of their own.

First select archives by single-clicking within the archive's grey title header. This will cause the archive to be outlined in blue, rather than grey (as below). If the comparison archive has been selected, it too will be outlined in blue. As an archive is selected, its place is marked with a blue line on the slider.

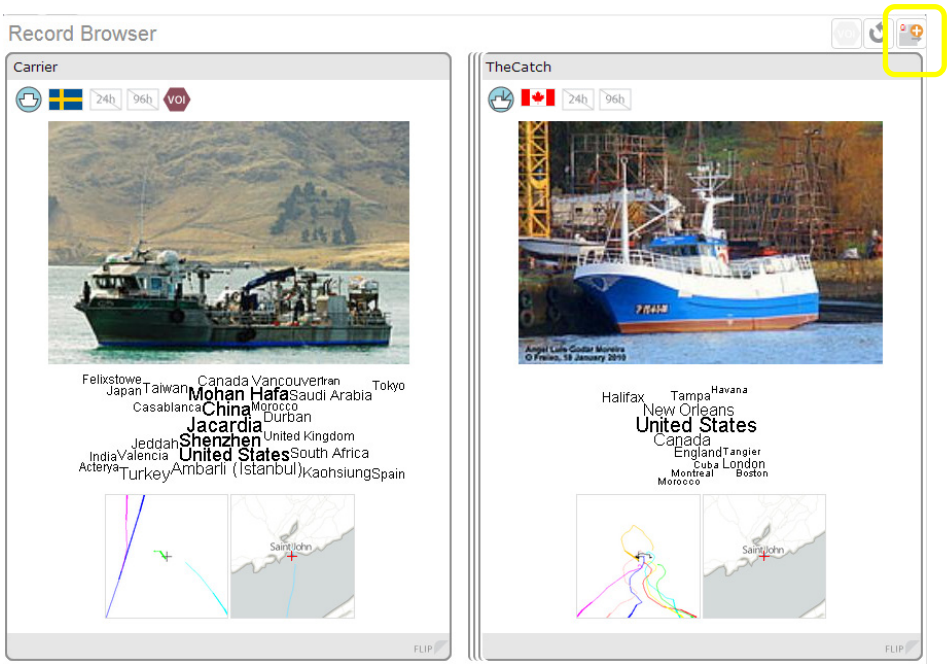


Figure 15: Record Browser: Export Ships
The Record Browser also allows users to select a subset of archives from a group, which can then be added to a new or an existing group.

You will also notice that as archives are added, the count increases in the Export Archives button in the top-right hand corner of the browser.

When the desired archives have been selected, press the 'Export Archives button and choose a Archive Group (Analysis Set) to add it to. The selected archives can be added to an existing set, or to a new set using the Add Archive Group button.

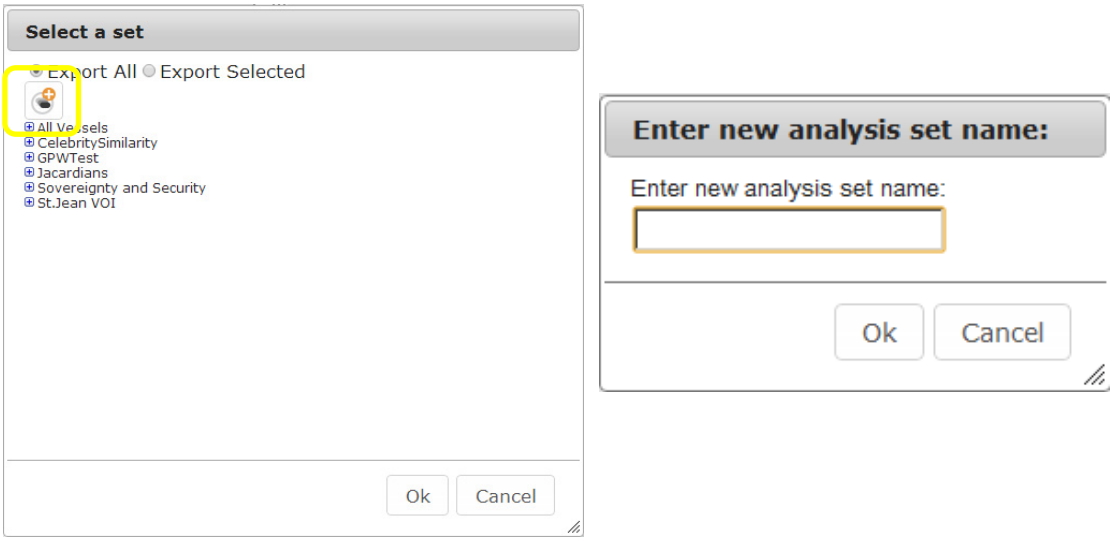


Figure 16: Export archives: Select analysis set, Enter new analysis set name
When the user selects to export archivess to a new group, they are prompted to select an analysis set (archive group) to add the selected archive to. The user can also select to create a new group using the Add Archive Group button. They will then be prompted to enter a name for the new group.

3.1.4.3 Mark as VOI

From the Record Browser, a user has the ability to mark the current ship as being a Vessel of Interest (VOI). To do so, the user can simply click on the VOI button (Mark as vessel of interest) in the top-right of the Record Browser widget area. Once a ship is marked as a VOI, the Vessel Summary Card itself will also be marked with a VOI icon.

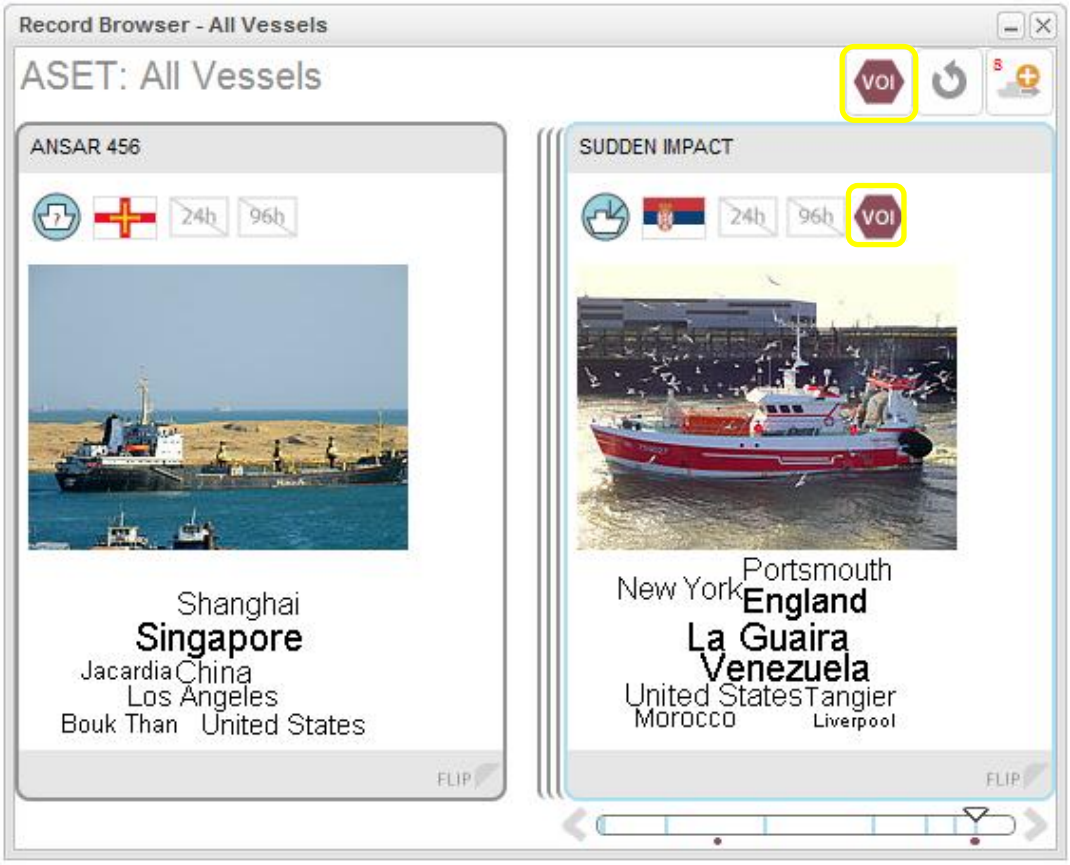


Figure 17: Record Browser: Mark as VOI
The user can mark a ship as being a Vessel of Interest from the Record Browser. Use the button in the widget's top-right corner.

3.1.4.4 Reset Selection

Once selections have been made within the Record Browser, these selections can be cleared using the "Clear the selection" button in the widget's top-right corner.

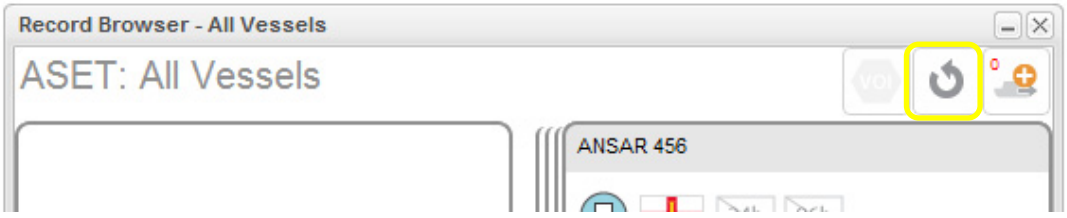


Figure 18: Record Browser: Reset Selection
Choose the button labelled "Clear the selection" to remove any selections made to the current set.

3.1.5 Timeline

The Timeline widget allows the user to visually examine reported events for ships of a Ship Group over a period of time.

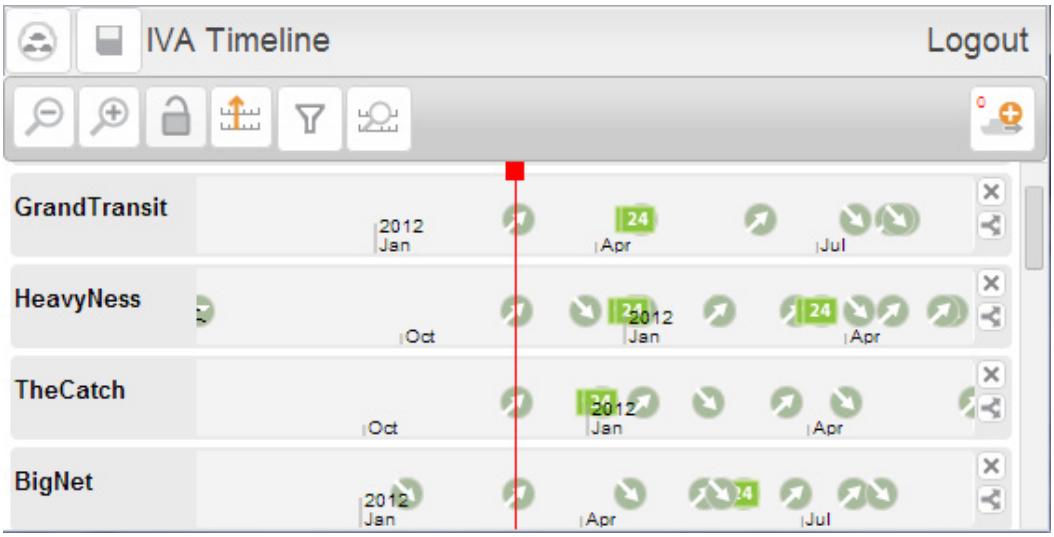


Figure 19: Timeline
The Timeline widget allows the user to examine reported events for ships of a Ship Group over a period of time.

Each ship is represented within a row, displaying events in chronological order from left to right. Events are of type 'Arrival', 'Departure', '96hr', '24hr' (call in to Canadian ports), 'Contact' and 'Sale of Vessel'. The user can mouse over an event to view its basic details. Double clicking on the event will synchronize events of the same type in all other rows to be aligned horizontally.

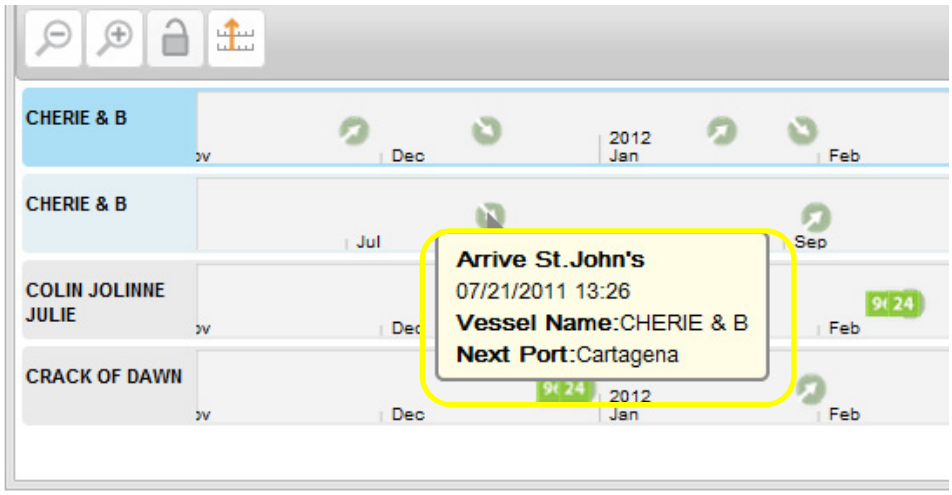


Figure 20: Timeline: Event Overview
Hover over any event in the timeline to view its basic details.

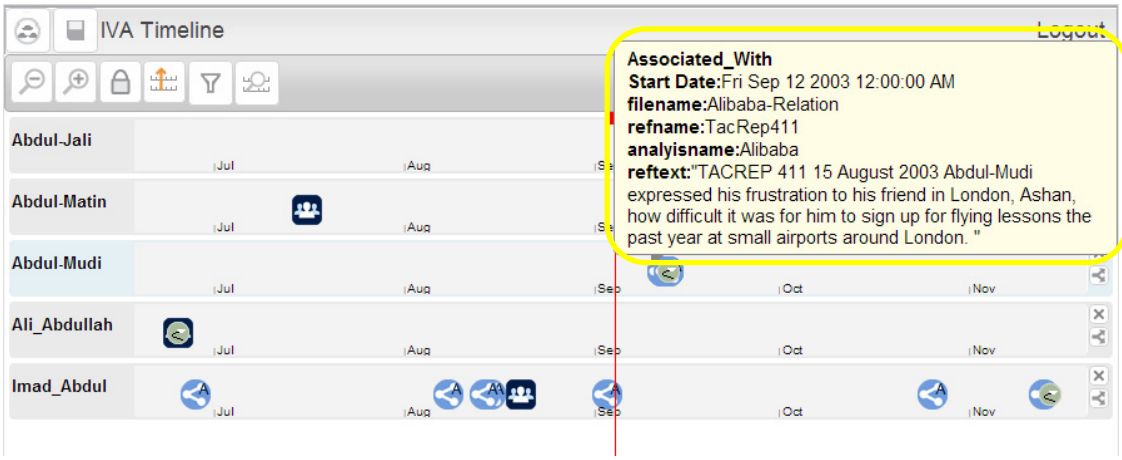


Figure 21: Timeline for Social Network Analysis: Event Overview

3.1.5.1 Timeline Controls: Zoom, Move, Lock & Unlock

The top bar of the Timeline shows some of the controls available in the widget. Zoom In ( ) and Zoom Out ( ) control the timeframe that is visible within the window.

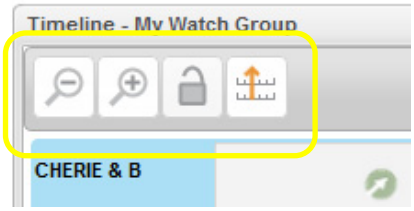


Figure 22: Timeline Controls: Zoom, Move, Lock & Unlock
Controls in the top-left corner of the widget allow the user to zoom in, zoom out, control local vs. global scrolling, and align timelines.

The user can also move the ships' timelines to either move forward or back in time using the mouse. Simply down-click within the timeline, and drag it either to the left (back in time) or right (forward in time). Release the mouse button to retain the new position in time.

The red line on the timeline can be repositioned by dragging the mouse. It controls the center point for the zooming operation.

The Lock control ( ) allows the user to determine whether all timelines move together (scrolling is global), or whether a single ship's timeline moves while the others stay stationary. While Lock is "on", use the mouse to drag a single ship's timeline.

Align Timelines ( ) aligns all the ships' timelines with the top-most ship.

3.1.5.2 Timeline Controls: Filter and Jump to Date

The Jump to Date button ( ) pops up a dialog allowing the user to manually enter a date to zoom in on.

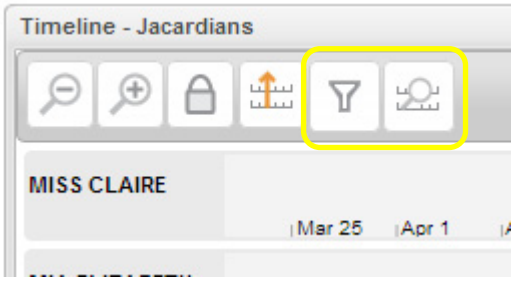


Figure 23: Timeline Controls: Filter and Jump to Date

The Filter button ( ) pops up the filter dialog. The filter dialog allows the user to manually adjust the time/region filter and to manipulate event type display. New types of events can be added by clicking the plus button at the bottom. The icon can be changed by clicking on the event icon and the event type can be deleted from the database of event types by clicking on the Delete button beside the event type.

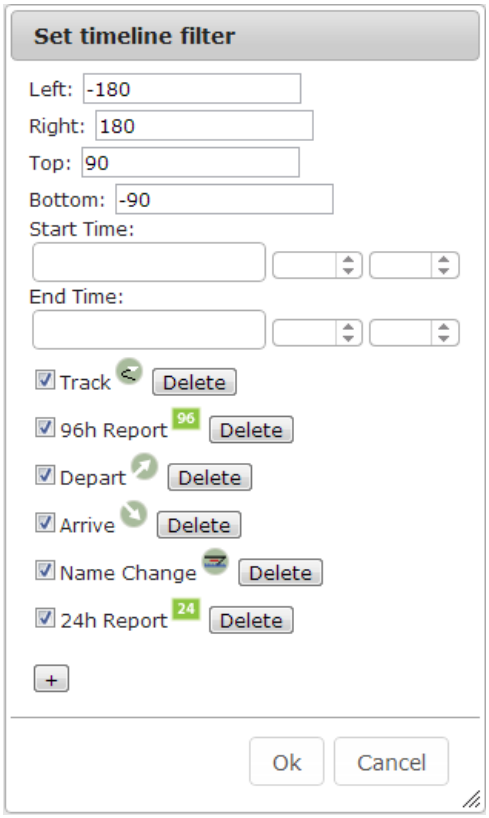


Figure 24: Timeline Controls: Set Timeline Filter

3.1.5.3 Remove Ships

Ships can be removed from or duplicated within the Timeline viewer with the control buttons at the end of each ship's row. The 'x' ( ) button removes a ship from the Timeline's view (though not from the group), the split button ( ) adds another instance of the ship to the Timeline's view. This can be useful to look at patterns in the ship's event history.

Figure 25: Timeline: Row Controls
Controls to the right of each row allow the user to remove or duplicate that row.

3.1.5.4 Export Subset

The Export Ships button here also allows the user to add a subset of ships from the timeline to a new or existing ship group. The functionality is the same as in the Record Browser, and as within the Record Browser, the icon is in the top-right corner of the widget window.

3.1.6 Magnets Grid

The Magnets Grid widget allows the user to visualize a wealth of information for groups of ships, with the intent of spotting irregularities of interest. Vessel attributes are used in the Magnet interactions, Dust properties, and Axis definitions to detect ship groupings.

Magnets Grid first opens with each vessel (represented by 'dust' particles) at the centre of the working area.

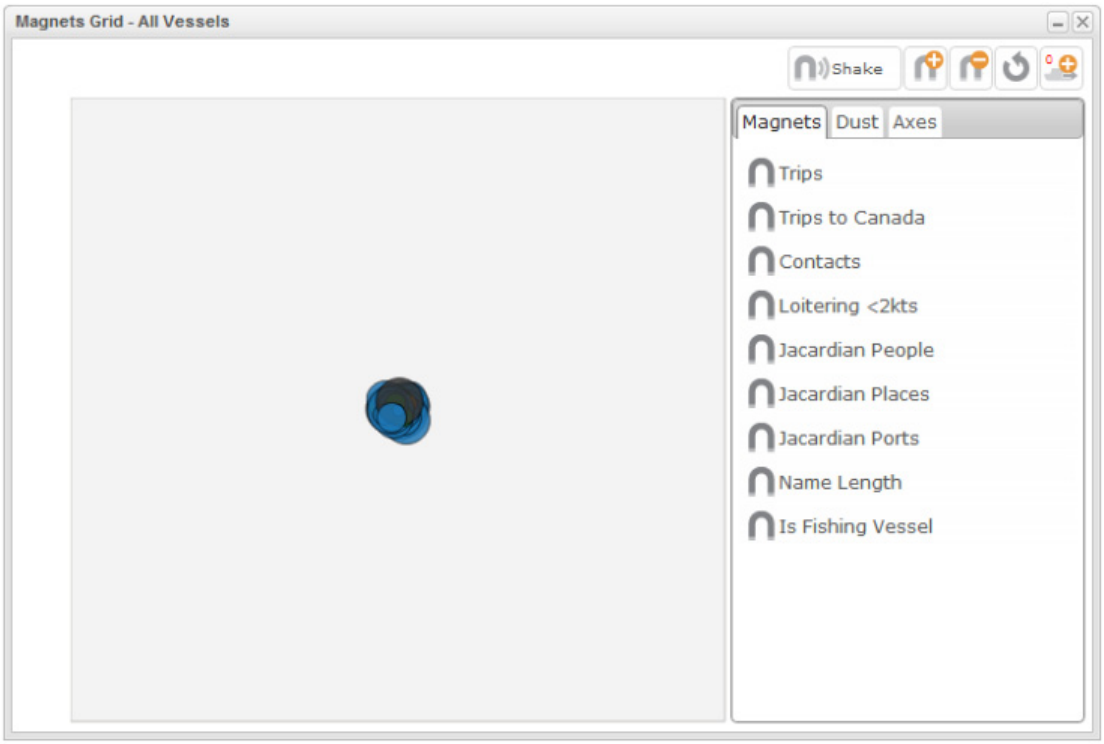


Figure 26: Magnets Grid
The Magnets Grid allows the user to visualize ship information with the intent of identifying irregularities of interest.

3.1.6.1 Attribute Customization

Attribute definitions may be created and deleted using the buttons circled in Figure 22. Creation is performed using the 'Add Attribute' button and entering a name, selecting the desired attribute field, and selecting whether it is categorical or numerical. For categorical attributes, the user can select which attribute values to include. Categorical attributes can be used to define Dust colours and Axis definitions. Numerical attributes can be used to define Magnets, Dust size, and Axis definitions. These can be removed using the 'Remove Attribute' button and selecting which attribute to remove.

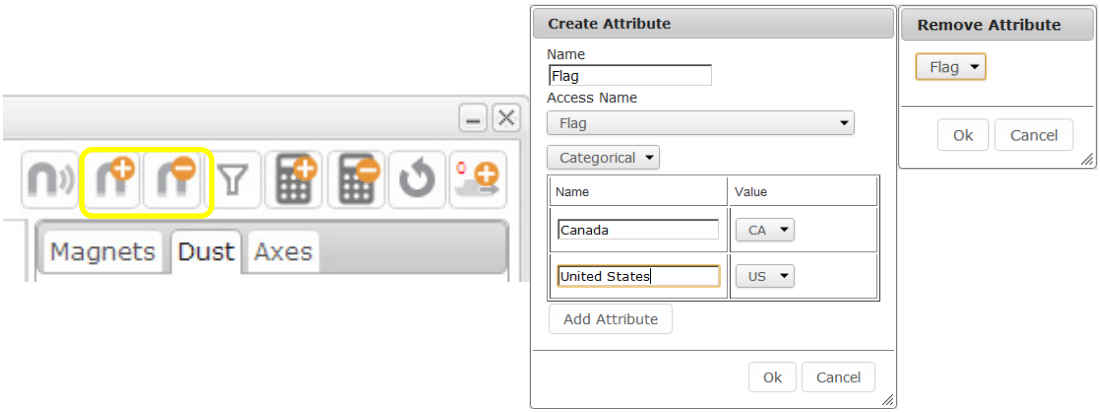


Figure 27: Magnets Grid: Attribute Creation and Removal
Magnets can be created and removed using the control buttons circled above. Dialogues shown are used to guide users through the creation and deletion process.

3.1.6.2 Calculated Attributes

Calculated attribute definitions may be created and deleted using the buttons circled in Figure 23. Calculated attributes count the number of events associated with an object that match a set of criteria and also match the time/region filter currently active for the magnets grid widget.

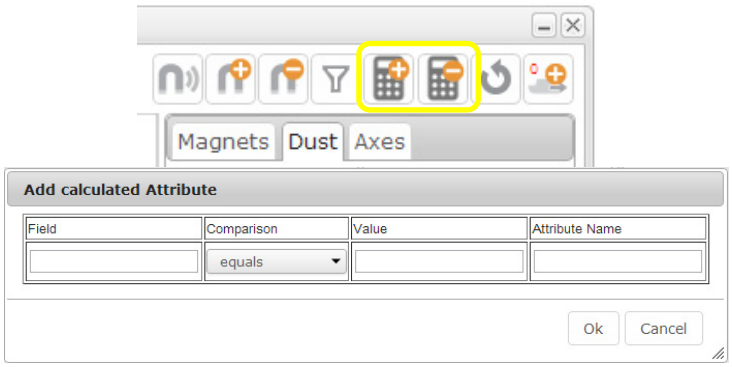


Figure 28: Magnets Grid: Calculated Attributes

3.1.6.3 Magnets

The user can drag and drop magnets from the Magnets tool tab to the working area. When activated, each magnet provides varying levels of "pull" to vessel 'dust' based on the data associated with that vessel.

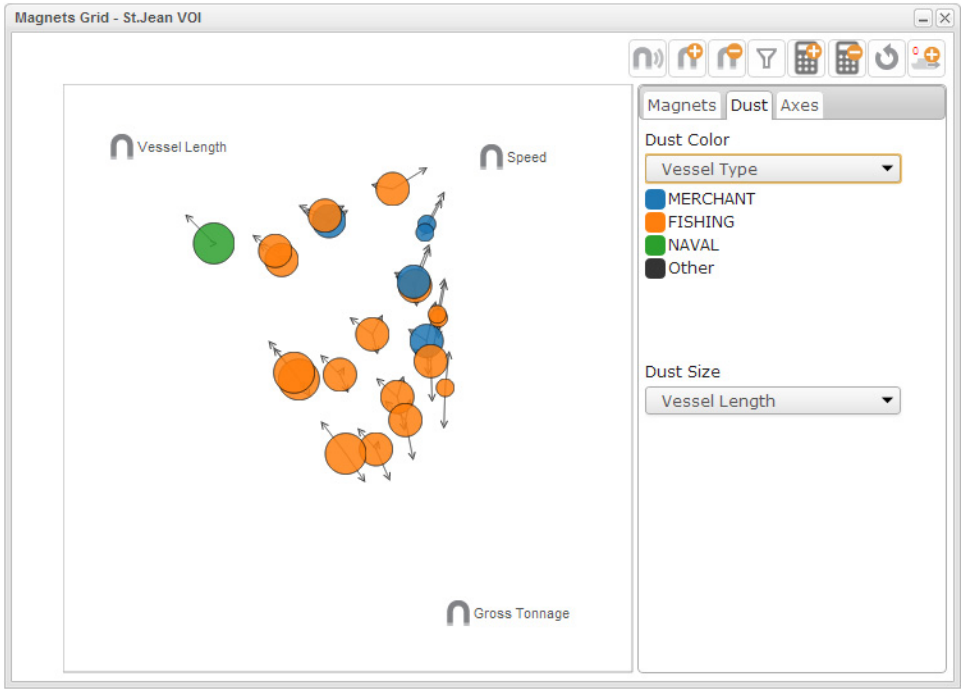


Figure 29: Magnets Grid: Magnets
Magnets on the Magnets Grid "pull" ships according to ship data. For quantitative magnets, the greater the quantity for that ship, the greater the pull from the magnet.

Hover over each magnet to reveal its controls. Activate each magnet with the 'shake' button . This will activate a pull on each vessel based on the vessel's values. The 'x' button removes the magnet from the working area.

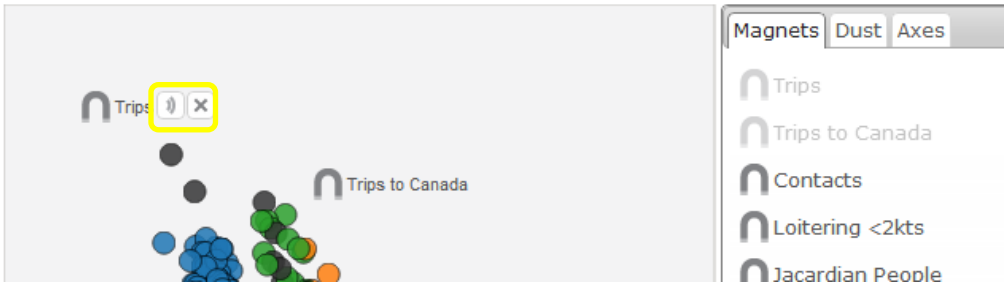


Figure 30: Magnets Grid: Magnet Controls
Hover over each magnet to reveal its controls.

3.1.6.4 Dust

The Dust tool tab allows the user to control the meaning of the Dust Colour, as well as of the Dust Size. Dust Colour can be chosen to represent the vessel type or (currently) any of the categorical or numerical magnet values.

Dust Size can also be chosen to represent any of the numerical magnet values (e.g. Number of Trips).

Arrows can be drawn showing the attraction between dust and magnets using the “show arrows” checkbox in the Dust tab.

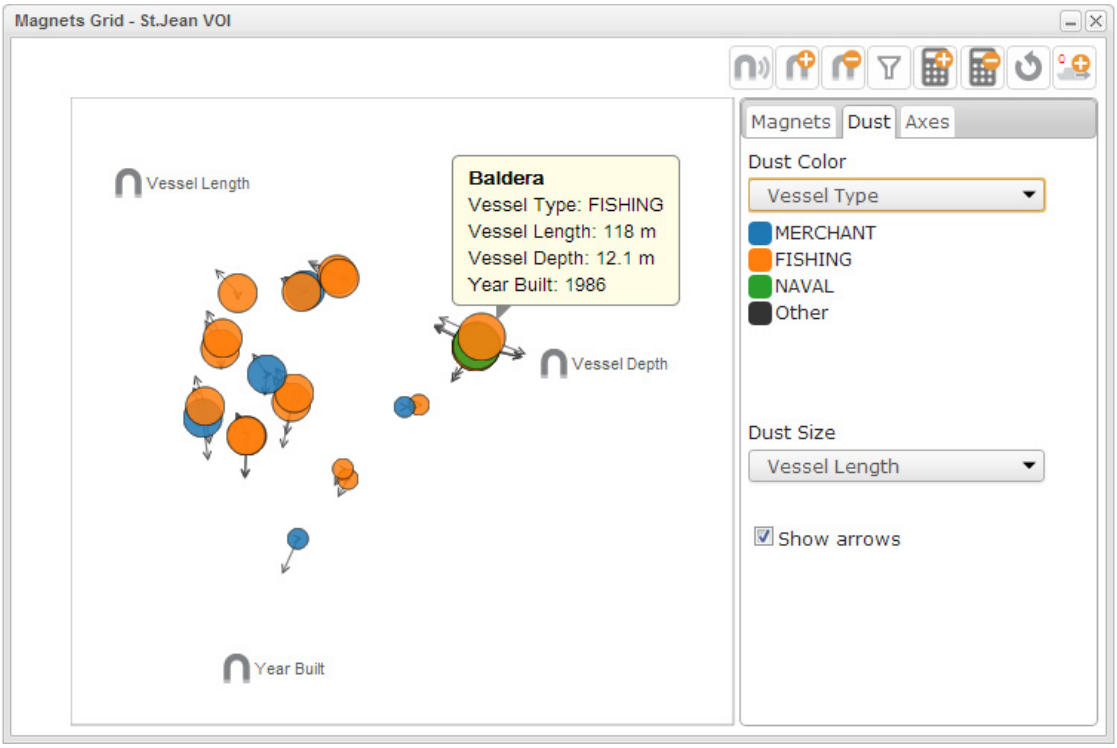


Figure 31: Magnets Grid: Dust
Each dust particle on the grid represents a ship. Dust colour and size can be changed to hold meaning. Single-click on a dust particle to reveal basic ship information.

As shown above, the user can get a read of basic information about each dust particle (ship) by single-clicking on it. This shows the ship name, vessel type, number of trips, etc.

As shown in the image below, numerical values are binned and colour is set to monotone within the same hue with a varying value.

3.1.6.5 Axes

The Axes tab of the tool box allows the user to specify static x and y-axes, which will plot the dust on a grid.

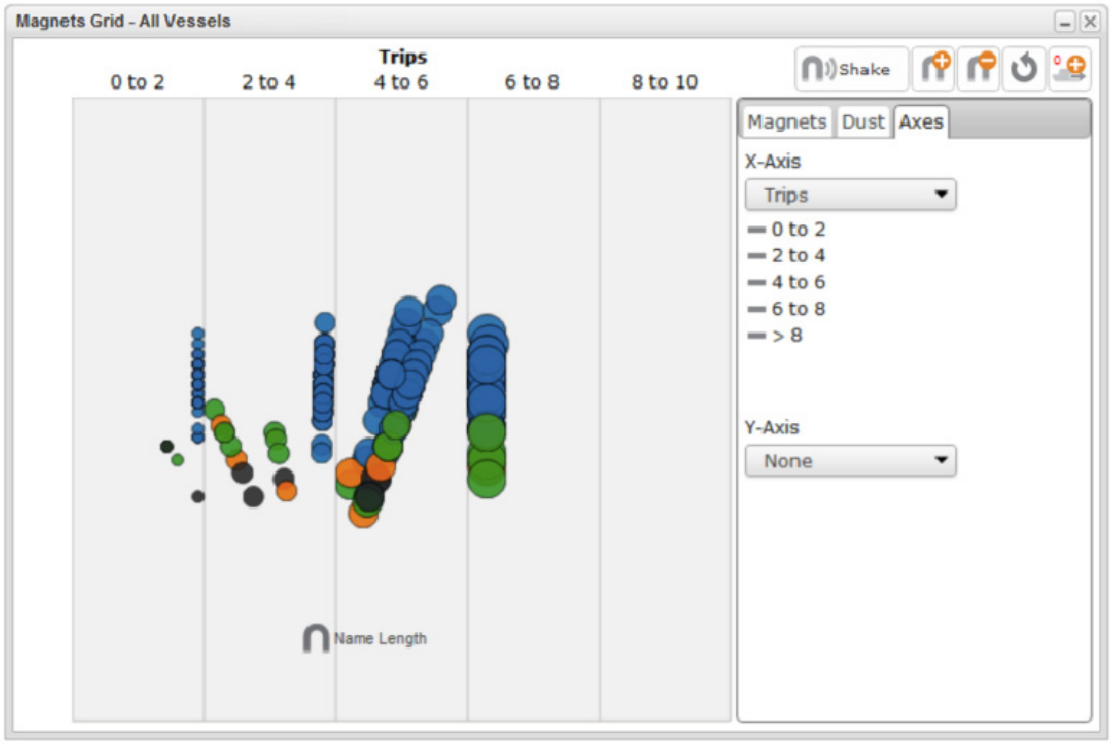


Figure 32: Magnets Grid: Axes
The Axes tab of the tool box allows the user to specify static x and y-axes, which will plot the dust on a grid.

A drop down list of all predefined attribute measures is provided for the x-axis (columns).

Axes can be used concurrently with Magnets. As without the axes, magnets added will affect the dust, but dust can only move within its axis boundaries (i.e. within the column).

When both the x- and y-axes are activated, dust is then further restricted and can move according to the magnet's pull only within the box created by the x- and y-axis.

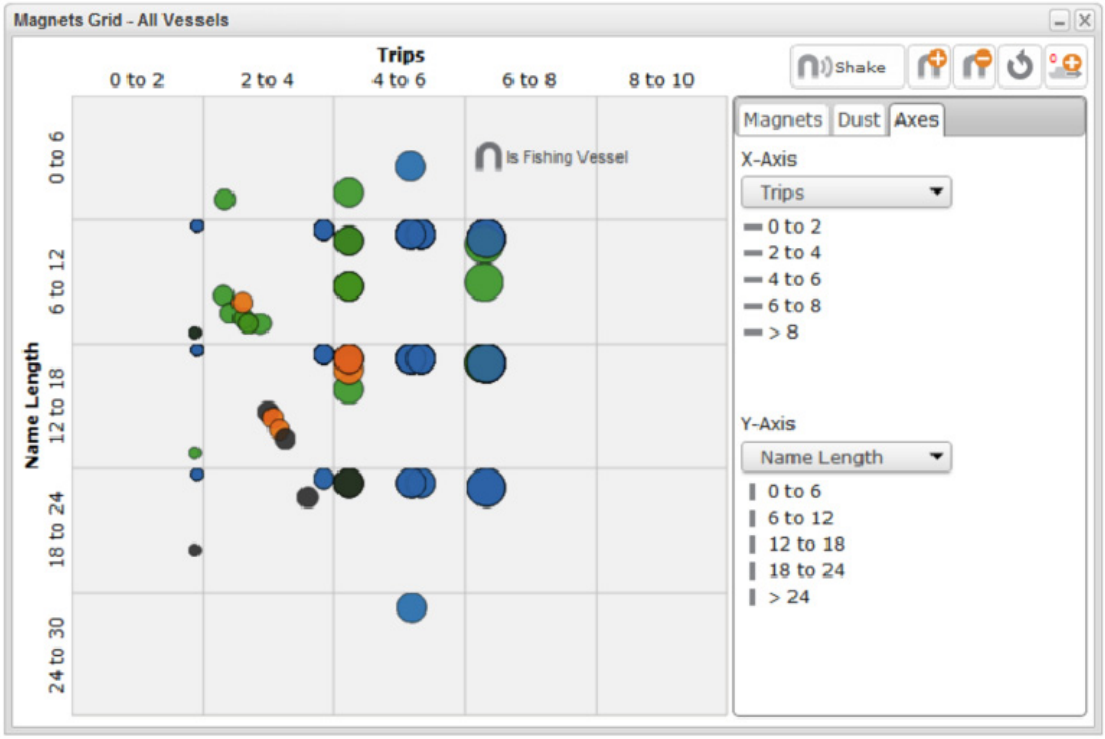


Figure 33: Magnets Grid: X and Y Axes
Both X- and Y-Axes can be added to the Magnets Grid.

3.1.6.6 Other Controls: Ship information, Select ship groups, Shake Magnets

Double-clicking on a dust particle (ship) opens the Vessel Summary Card for that ship.

Dust can also be selected in groups using the mouse. Simply click down on the mouse and drag it to create a square selection area. Let go to select all ships within the area. Add or remove ships from a selection by holding down the CTRL key while clicking on individual ships. A selection of ships can then be added to a Ship Group (Analysis Set) with the 'Add to ASET' button in the top right corner of the widget. Shake Magnets activates all magnets in the working area at once.

3.1.6.7 Standalone Magnets Grid

As with all the widgets, Magnets Grid can now be used as a stand alone web page outside of WAS.

To support standalone deployment it is possible to drag and drop a csv file into magnets grid. The first row of the csv file should be column headers which include "NAME" and "ID". Other column headers will be used to create attributes for use in magnets, axes, color and size affordances.

A standalone Magnets Grid session will automatically be assigned a sessionid in the URL. The save button will store the sessionid along with all of the session state including dust data and positions. Thus, it is possible to bookmark the session or instantiate it in SitScape or OWF such that the session can be restored exactly on reload.

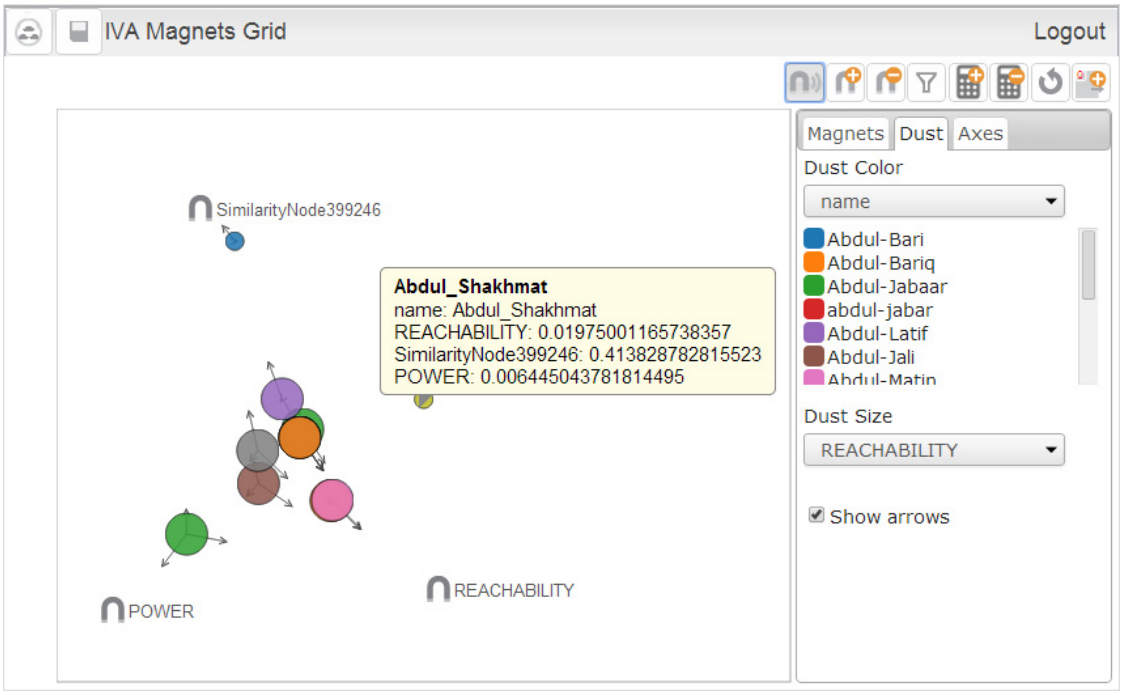


Figure 34: Standalone Magnets Grid
Magnets Grid as a standalone deployment, used here for social network analysis.

3.1.7 Map and Timeline

The Map and Timeline widget allows the user to interactively view vessel positional data.

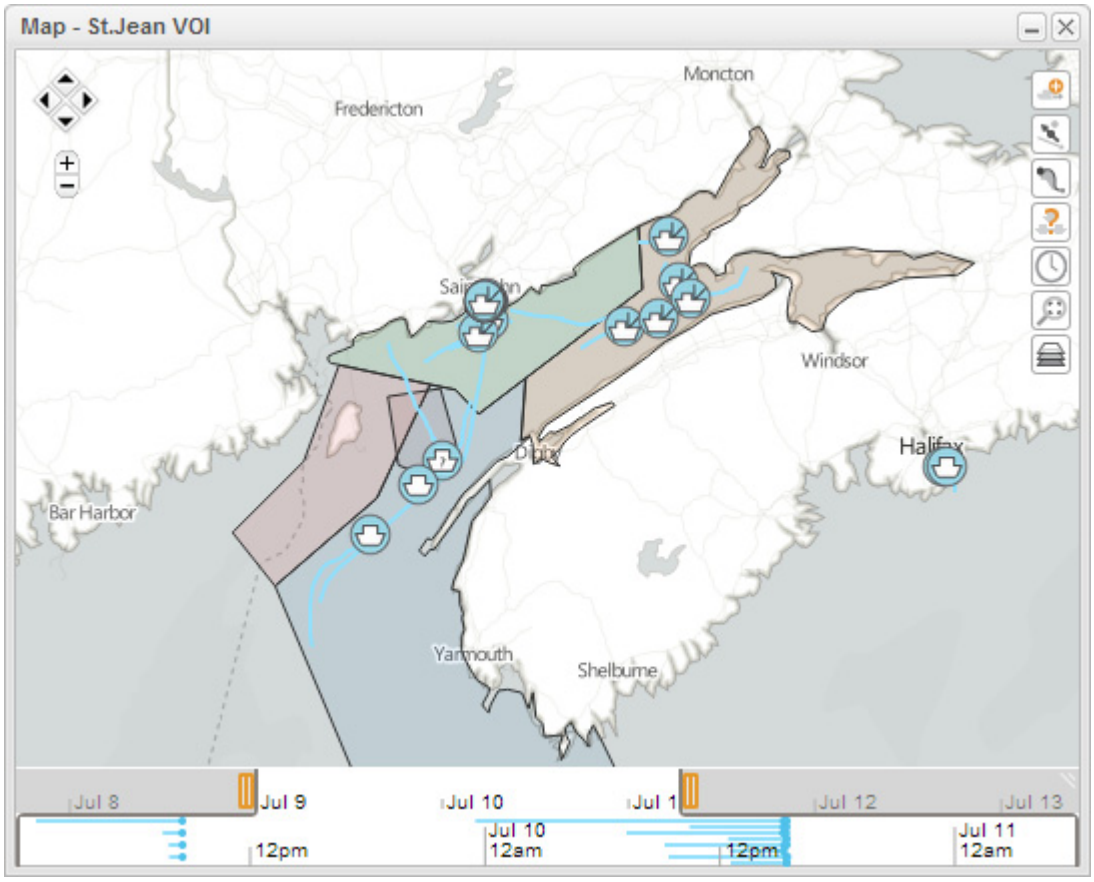


Figure 35: Map and Timeline
The Map and Timeline widget allows for detailed analysis of historical vessel contact data.

The timeline control under the map allows the user to select a time window for which tracks will be visible on the map. The first row of the time control displays the active time window in a larger context. The second row displays only the active time window plus bars to indicate recorded tracks for vessels. The time bars can be zoomed independently using the mouse wheel or panned with a mouse drag. As the time window is adjusted, the map display will animate. The time control can be resized by dragging in the top right. Note that when using the mouse wheel to zoom the map or timeline, the point under the mouse cursor remains fixed.

The map display shows the vessel track for the active time window plus a vessel icon at the interpolated vessel position at the end of the time window. Vessel information can be displayed in tooltips by moving the mouse over a vessel or track. Clicking on a vessel or track will select the vessel.

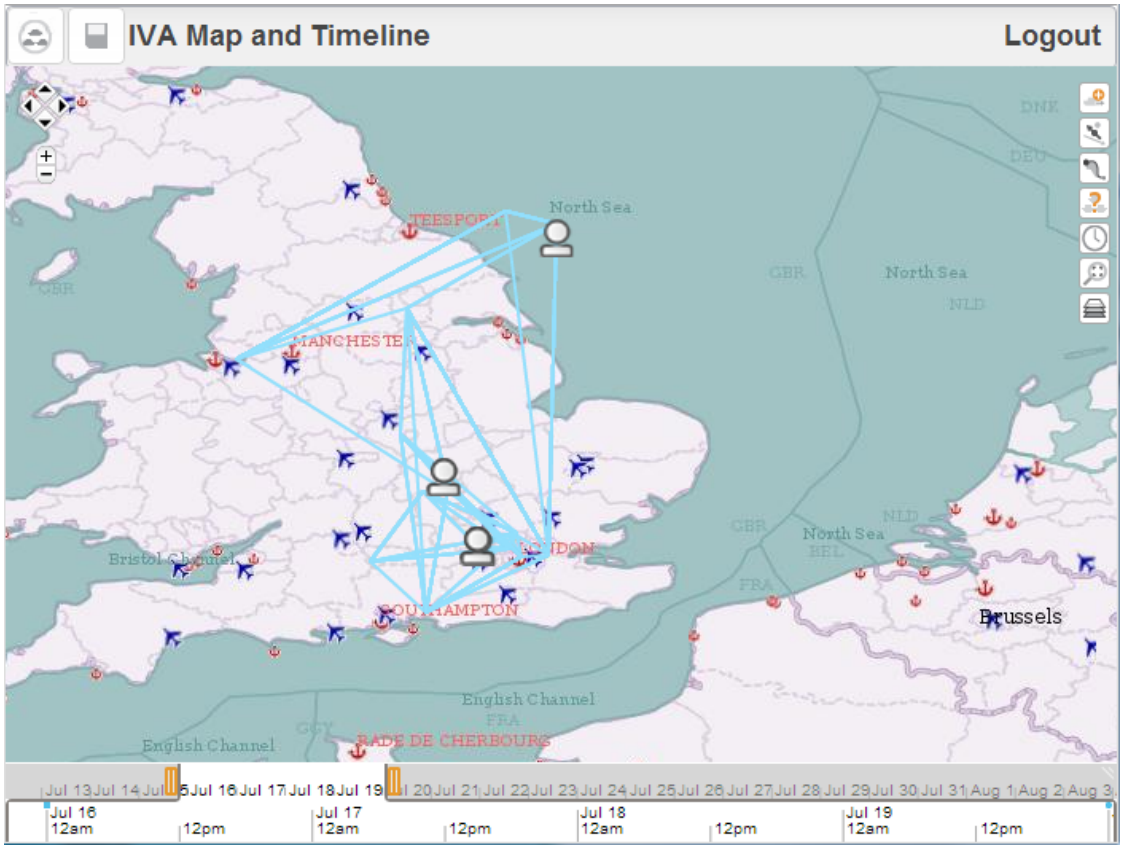


Figure 36: Standalone Map and Timeline
Map and Timeline as a standalone deployment, used here for social network analysis.

3.1.7.1 Close Encounters

Selecting a vessel results in the display of a Close Encounters Pop-up. Clicking on the close encounters icon will display these pop-ups for all vessels which have close encounters. Empty Close Encounters Icons are not displayed.

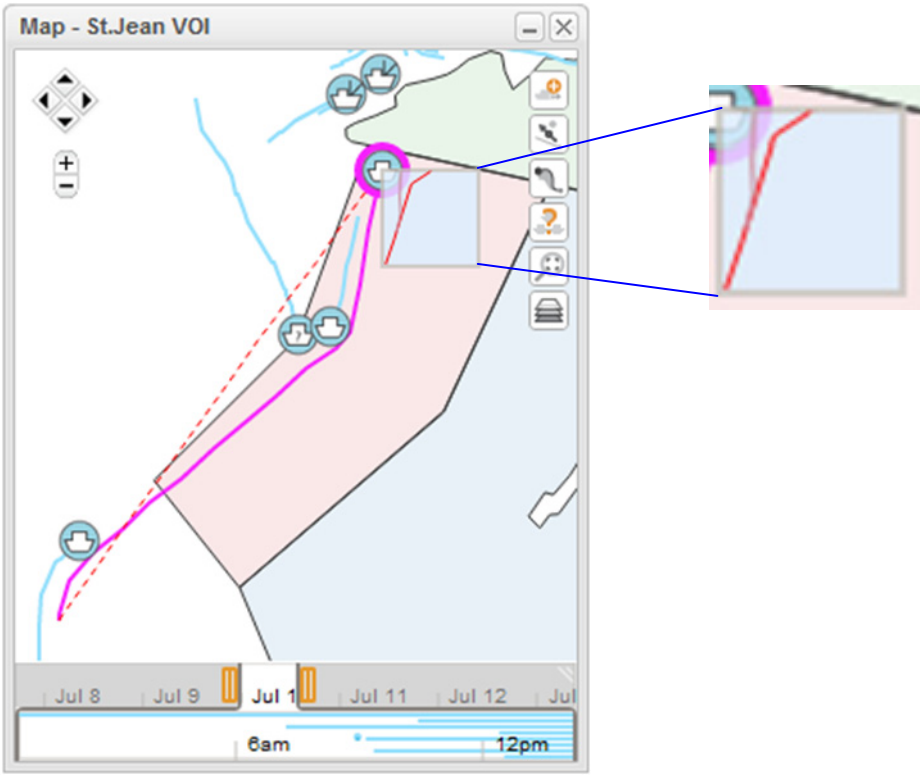


Figure 37: Close Encounters
The Close Encounters Pop-up displays vessel tracks that have been in close proximity. In this example Close-Encounters pop-up, expanded on the right, the red line indicates that another vessel has manoeuvred around the highlighted vessel, passing on its port side.

The Close Encounters Pop-up is rendered by plotting the tracks of other vessels relative to the position of the selected vessel. A virtual box is placed around the vessel and moved along the vessel's track. When other vessels enter the virtual box, a track line is placed accordingly. Using this approach it can rapidly be determined if other vessels have interfered with the selected vessel.

Close encounters that occur within the active time window will be displayed bright red. Encounters that occurred outside the active time window are faded.

3.1.7.2 Route Ribbons

Selecting a vessel results in the display of a dotted red line called a Route Ribbon. Clicking on the Route Ribbon button displays route ribbons for all vessels.

The route ribbon is a shortest path (great arc) between the vessel position at the start of the active time window and the vessel position at the end of the active time window. It can be used to identify if a vessel has diverted from an optimal path between two points which may indicate suspicious behaviour.

3.1.7.3 Map Controls

The Fetch Vessels button sends a request to the server to retrieve all vessels with tracks that intersect both the current active time window and the current visible map region. All other vessels will be removed from the display and replaced by this new set of vessels.

The timeline button launches the timeline widget with the same set of vessels with events filtered on the visible time/region window. The View Fit will center the tracks for the selected vessels or for all vessels if none are selected. The Layers button allows toggling the visibility of vessels, KML layers, and time bars. Vessels may also be selected using the layers dialogue.

3.1.8 Graph Analyzer

The Graph Analyzer widget allows the user to interactively view links and relations between archives

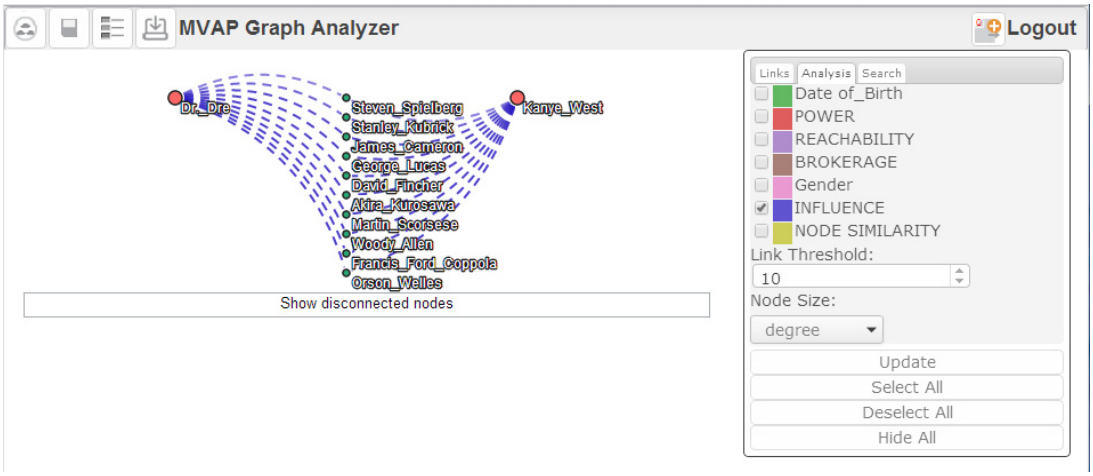


Figure 38: Graph Analyzer
The Graph Analyzer Widget allows the user to explore relations between archives. Interactive link filtering and related link operations are supported.

Data is loaded by opening the analyzer with a selected set from the Analysis Set Manager. When the widget loads, it will show any links/relations between the archives in the given set and display them as a node-link graph. All the link types contained in the graph are shows in an interactive legend on the right-hand side of the widget.

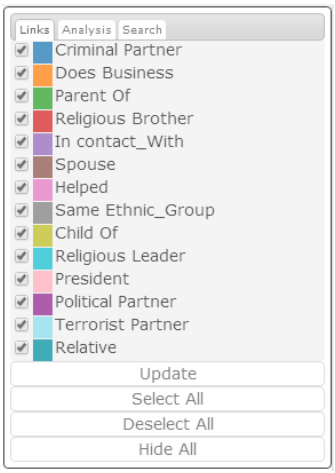


Figure 39: Interactive Legend
The interactive legend allows for real time filtering of link types in the Graph Analyzer

The legend contains three tabs: Links, Analysis, and Search. The functionality of each is explained below.

3.1.8.1 Links Tab

The Links Tab in the interactive legend displays a checkbox and a name for each link type. If the link type is checked, it will be included in the displayed node-link graph. The Select All button will mark all link types as included, while the Deselect All button will mark all link types as excluded. The Update button will re-draw the graph with the given subset of link types. The Hide All button will keep the position of each node in the graph and hide any links that exist between the nodes.

3.1.8.2 Analysis Tab

The Analysis Tab displays any Analytic Links between nodes/archives. Analytic Links are values between archives that are given as output from the Network Analyzer Service. We display these values as links in a graph. To visually denote the difference between Standard Links and Analytic Links, we use a solid line for Standard Links and a dashed line for Analytic Links.

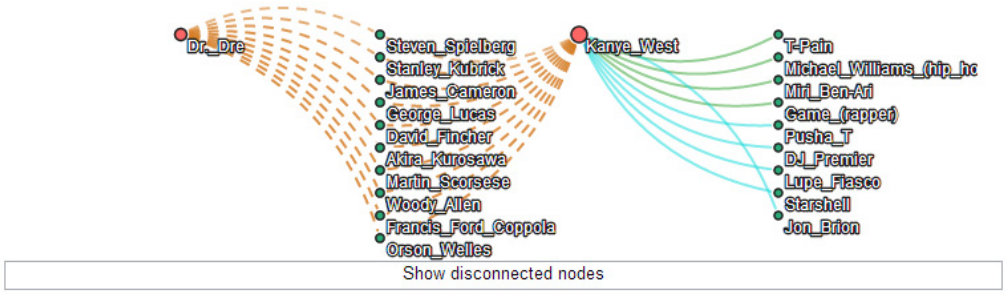


Figure 40: Standard and Analytic Links
All links to the left of Kanye_West are Analytic Links and shown as dashed lines. All links to the right of Kanye_West are Standard Links and shown as solid lines.

The Analysis Tab also allows us some extra control over the number of links shown in the graph. It would be undesirable to show all Analytic Links for a given graph. For each node, there is an Analytic Link to every other node in the graph for each analysis metric that came from the Network Analyzer Service. For this reason, we have a Link Threshold control that allows to display the top N Analytic Links for a given type.

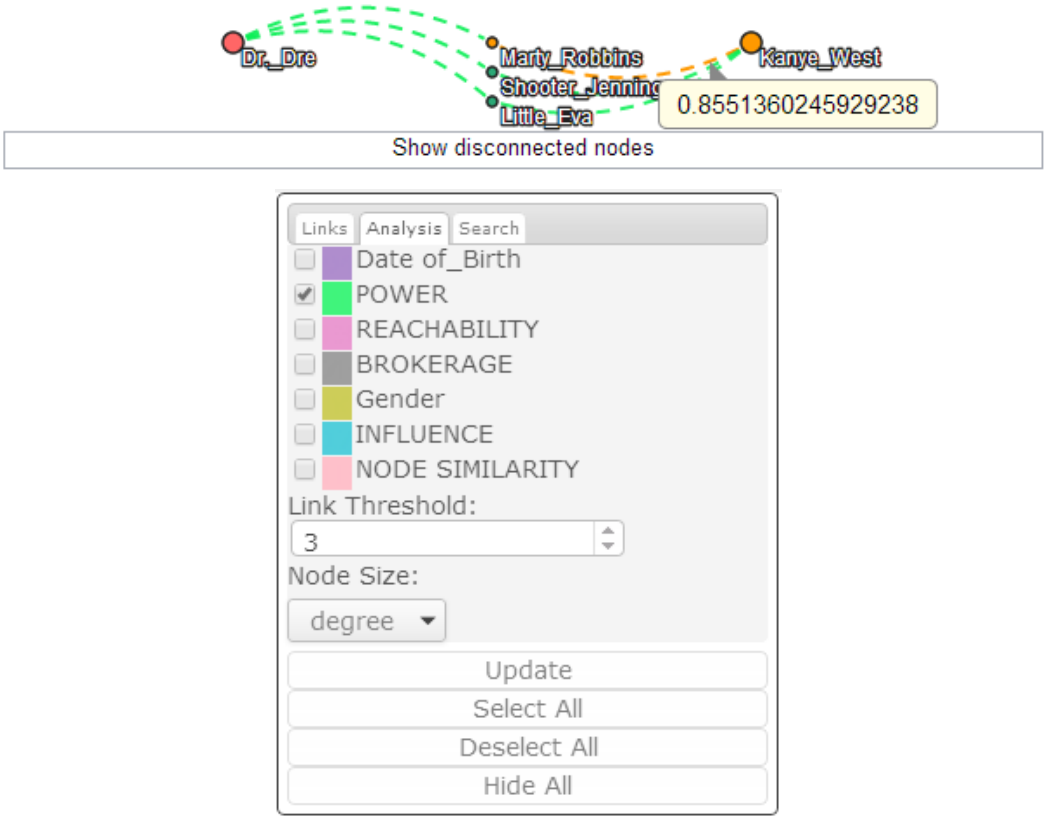


Figure 41: Analytic Link Threshold
We show a graph for the Analytic Link "POWER". The top 3 results for each node are shown.

We see above the Link Threshold is set to three. Thus, the top 3 Analytic Links for the given type ("POWER" in this case) will be shown for each Similarity Node. To aide in identification, Similarity Nodes are highlighted with a red fill.

3.1.8.3 Search Tab

The Search Tab is used to search for a text string in the graph. It searches the labels on the nodes and highlights any result in red. This functionality is similar to the standard search functionality available in your browser, but is tailed specifically to the Graph Analyzer. If the next result is off-screen, the widget will automatically center (or center as much as possible) the current search result.

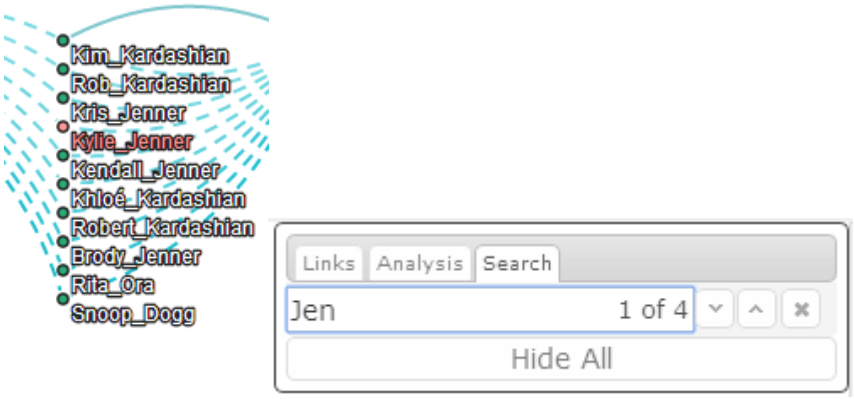


Figure 42: Graph Searching
An example search for the string 'Jen'. To the left, the current search result is highlighted in red.

3.1.8.4 Disconnected Components

At the bottom of each graph is a button with the text "Show/Hide Disconnected Nodes". This will toggle the view of any archive in the set that is not connected by the filtered link types.

Figure 43: Disconnected Components
Disconnected components are arranged in a columnar format below the main graph. Click the button to toggle the viewing of disconnected components.

3.1.8.5 Node Selection

As in Magnets Grid, we support marquee selection for nodes in the Graph Analyzer. Hold the Shift key to add nodes (click nodes or marquee drag), or the Control key to toggle selection. Selected nodes are highlighted blue.

3.1.8.6 Context Menu

We can right click any node in the graph to perform various operations on the graph.

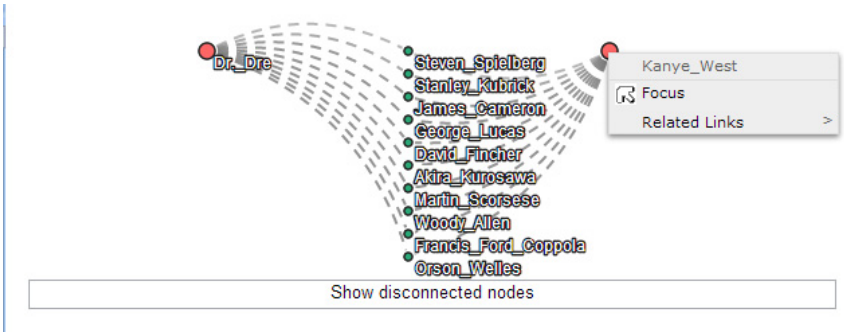


Figure 44: Node Context Menu
Right-clicking a node in the graph will activate a context menu for that node.

The "Focus" button will move that node to the top-left corner of the graph. The "Related Links" submenu will show a list of all available link types in the given analysis. We can click a link type to fetch all nodes of that type that are connected to the given node. We search any input graphs for the current analysis links pertaining to this node.

3.1.8.7 Toolbar

The toolbar for the Graph Analyzer contains some functions that are unique to this widget.

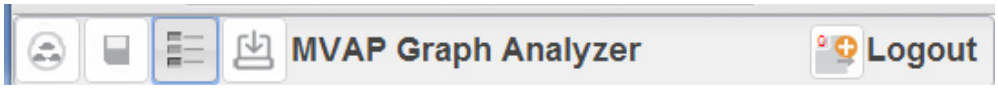


Figure 45: Graph Analyzer Toolbar
The buttons from left to right: Return to Analysis Set Manager, Save State, Show/Hide Interactive Legend, Import Graph Data, Export To Analysis Set.

The first two (Return to Analysis Set Manager, and Save Session are similar to the other widgets. The Show/Hide Legend button will add or remove the Interactive Legend from view. The Graph Import Button allows us to import data defined in GraphDTO format into the system.

3.1.8.8 Importing Graph Data

We allow the user to import data specified in GraphDTO format through the client itself. To import data, click the "Import Graph Data" button from the Graph Analyzer Widget.

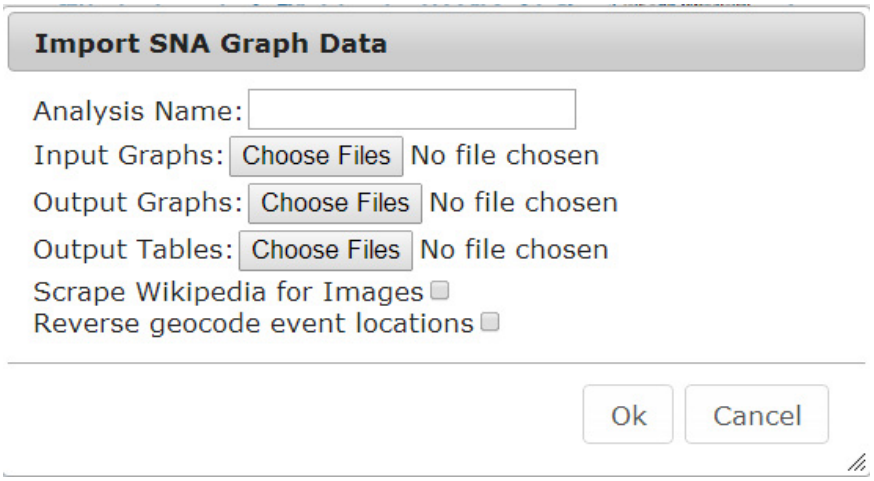


Figure 46: Graph Import Dialog
The dialog form for importing SNA graph data into the system.

The data is imported as follows. We must specify a name for the given analysis. (Eg. CelebritySimilarity). This will allow us to associate any subsequent graphs and tables to the given analysis. The "Input Graph" row allows us to specify GraphDTO files that were used as input to the Network Analyzer Service. The "Output Graphs" row allows us to specify GraphDTO files that were produced as output from the Network Analyzer Service. The "Output Tables" allow us to specify NodeBasedResults produced by the Network Analyzer Service. The "Scrape Wikipedia for Images" checkbox will attempt to locate an image for each imported archive on Wikipedia. The "Reverse geocode event locations" will find the closest major city for each latitude/longitude pair for event locations associated to each archive. This is used to enrich the data in the word cloud that appears on the Visual Summary Card for each archive.

When the data is imported into the system, you will notice a few automatically generated Analysis Sets have appeared. The system uses a combination of the Analysis Name specified, plus the filename/type for each file to determine how links are organized between the archives in each Analysis Set. For example, if we use "CelebritySimilarity" as the analysis name and specify two input graphs: "EconomyEthnicityEducationReligionGraphDtoStandard" and "EconomyTribal_AfiliationGraphDtoStandard", one output graphs: "ECONOMY_ETHNICITY-#Helped" and three output tables: "AttributeSimilarity-ECONOMY_ETHNICITY-#Helped", "NodeSimilarity-ECONOMY_ETHNICITY-#Helped", "SingleSimilarity-ECONOMY_ETHNICITY-#Helped", we would see the following autogenerated Analysis Sets in the Analysis Set Manager:

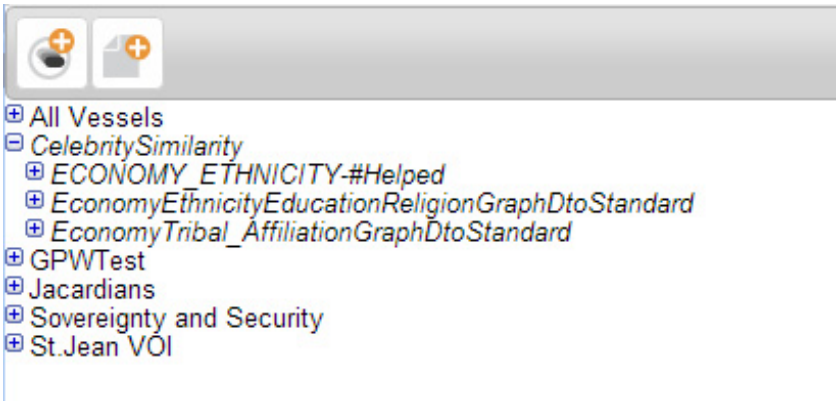


Figure 47: Autogenerated Analysis Sets from Importing Graph Data

Notice that we have a root Analysis Set that corresponds with the Analysis Name provided in the Graph Import Dialog. Contained in that Analysis Set is a set for each input graph specified. We also have a set generated for each output graph specified. The Analytic Links defined in the NodeBasedResult tables are associated with the output graph that corresponds to that specific relation.

3.1.9 Standalone deployment

In Phase 3 it was made possible to deploy the apps outside of the WAS framework. Building the MVAP_UI project and proceeding to the http://<tomcatserver>/MVAP page will display all of the widgets that can be launched independently. The timeline widget is not present since it must be launched with a set of vessels from either the analysis set manager or the map and timeline. Once the widgets are started in standalone mode, they behave much the same way as when deployed within the WAS except they occupy the entire browser window.

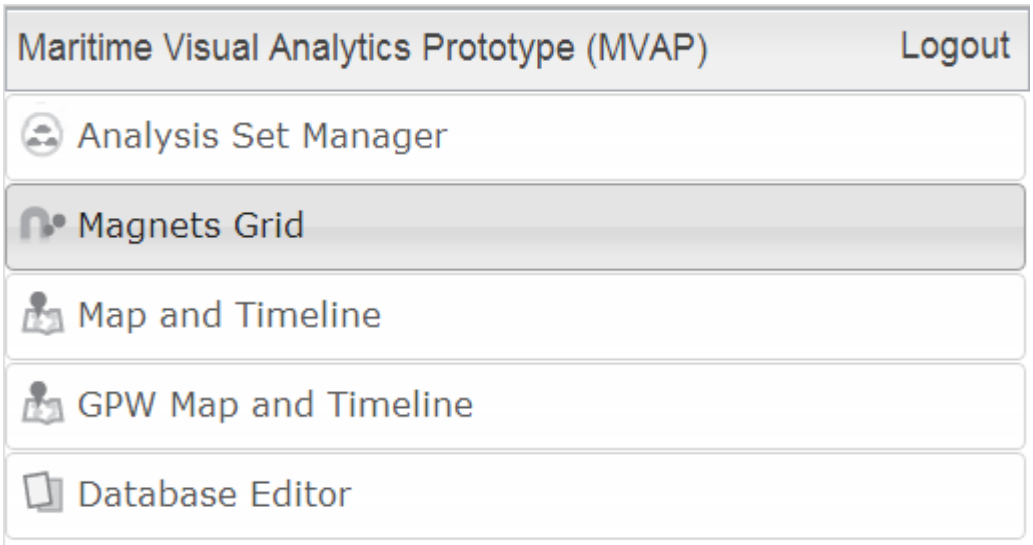


Figure 48: Standalone MVAP Main Page

3.1.10 GPW Map and Timeline

The GPW Map and Timeline widget was created as a better way to view a massive volume of GPW data.

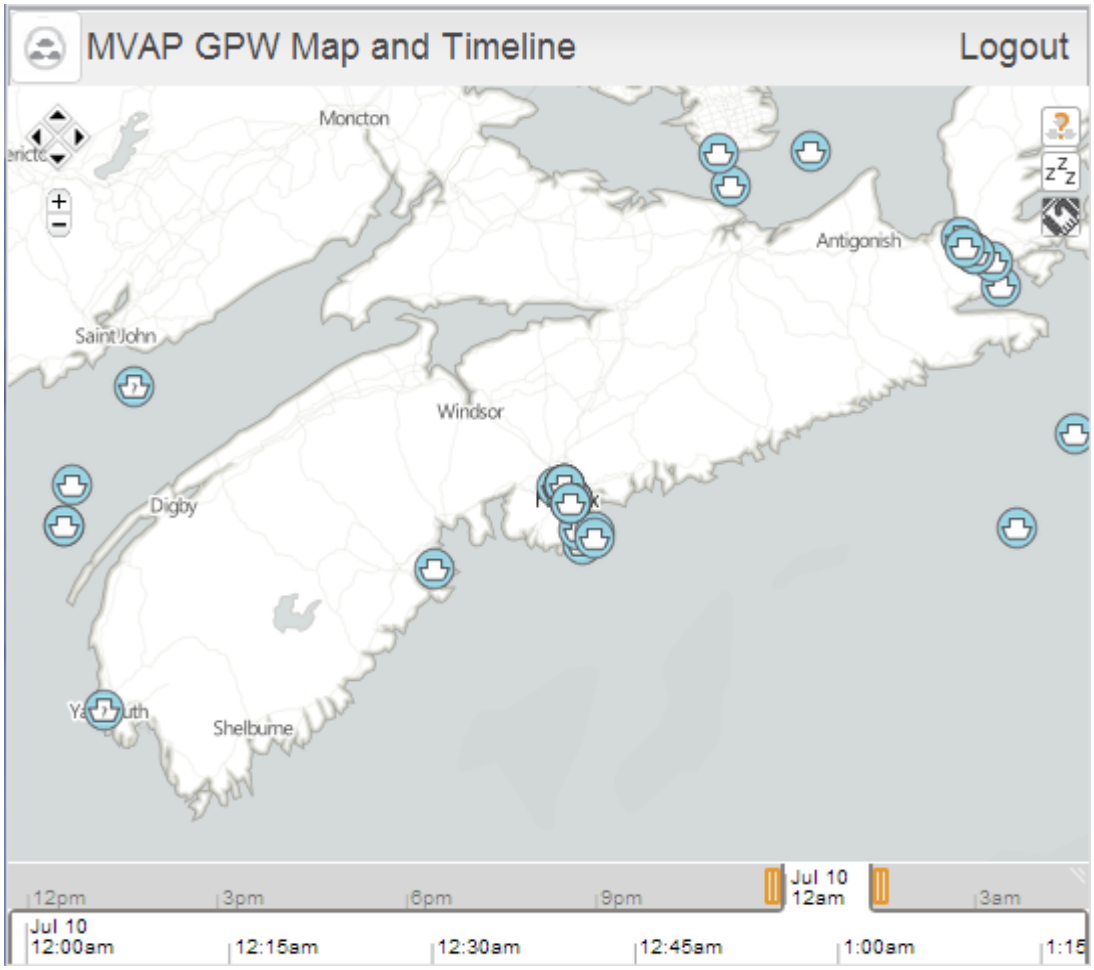


Figure 49: GPW Map and Timeline

The display starts with a blank map and the user should zoom in to a time and region of interest before clicking the "Fetch Vessels" button at the top right. Selecting a region that is too large will result in poor performance. The icons displayed by fetch vessels represent the latest position of vessels that have reported within the selected time/region window. As an alternative to "Fetch Vessels" the user could choose to "Fetch Loiterers" or "Fetch Encounters" which would fetch vessels loitering or having close encounters within the selected time/region.

After fetching an initial set of vessels, the user can mouse over the vessels to view whatever details are available. In the case of loiterers and encounters, these details will be limited. Clicking on a vessel will produce a menu with options to fetch more data for the vessel or to export it.

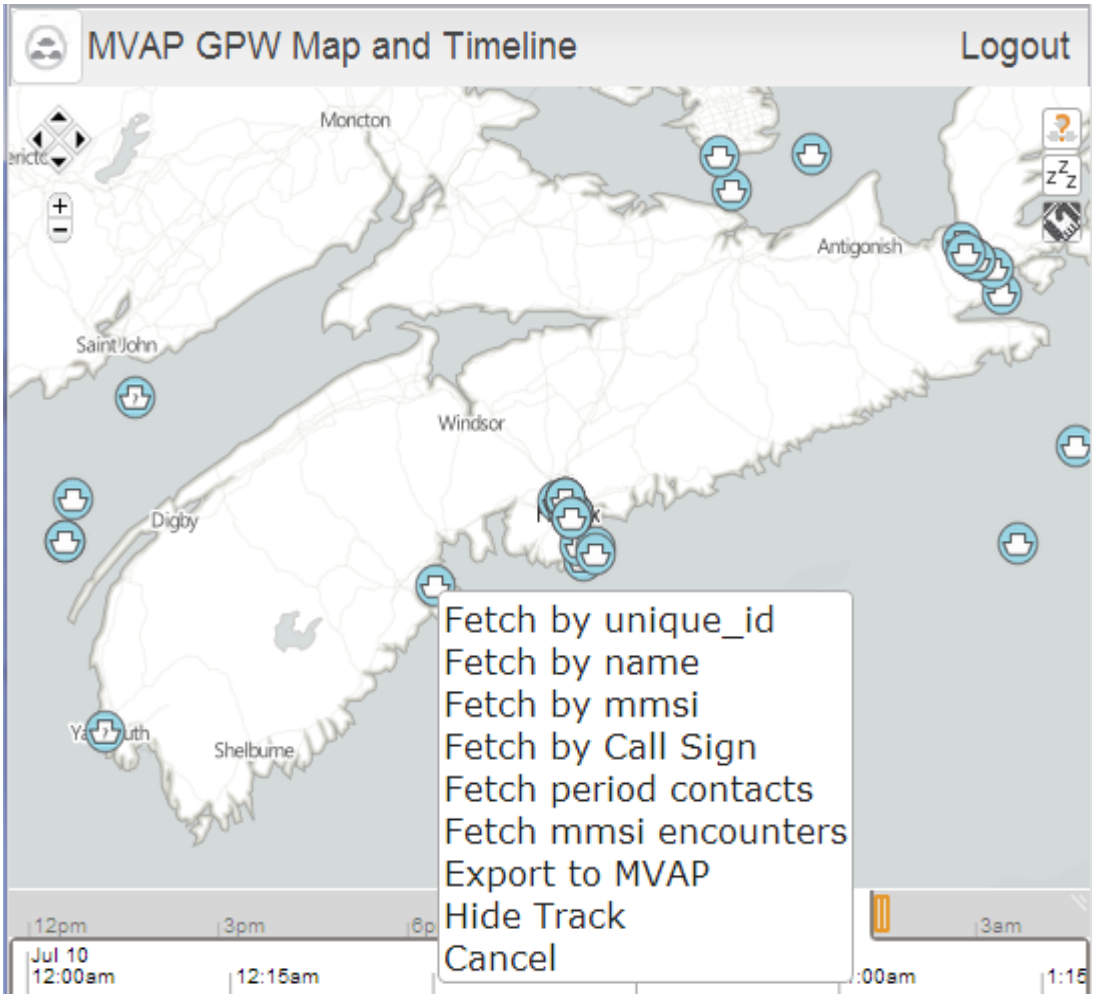


Figure 50: GPW Map and Timeline Context Menu

The Fetch options in the menu will retrieve tracks for the vessel matching the given field. The tracks will be constrained to the selected time window. Note that the vessel may move since the position will be animated and controlled by the time slider after the track is fetched.

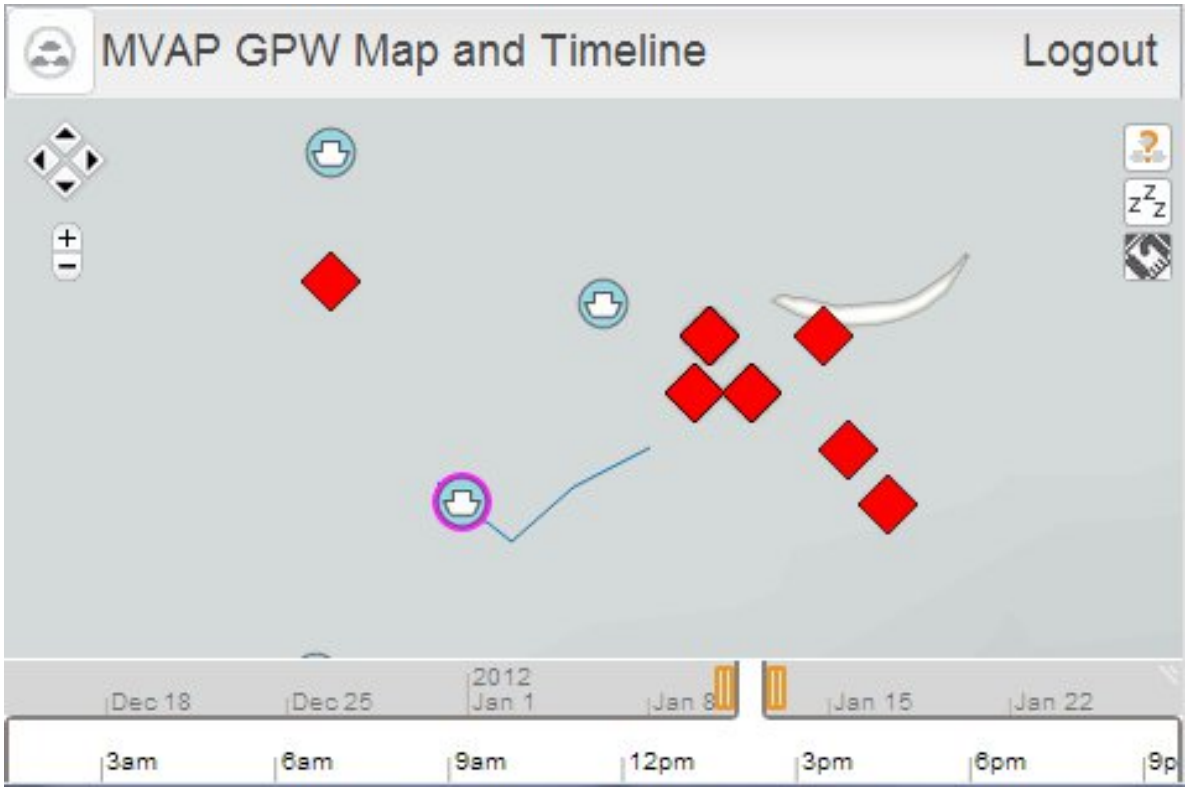


Figure 51: GPW Encounters

Selecting the export to MVAP option will open the export dialog which will allow the user to automatically search shipspotting.com for more data and imagery for the vessel. The vessel data and tracks can then be exported to the main MVAP database.

3.2 Design Feedback Not Implemented

During MVAP development multiple software drops were delivered to DRDC for installation in the Information and Intelligence Section environment. A hosted server was also stood up at Oculus with a guest account provided for remote access. User feedback was obtained, and through an iterative process each software drop addressed bug reports and most feature requests were implemented.

Design feedback that was not within the scope of effort is captured in the subsections that follow.

3.2.1 Analysis Set Manager

- Go directly to the vessel selection after creating a new ASET

3.2.2 Vessel Summary Card

- "Can there be clickable links?"
  - Yes, there already are some; need to discuss more uses for links as time allows
- Vessel IMO on front

3.2.3 Timeline

- Events that have a timespan:
  - "You could add an icon showing the flag of the port between the arrive and depart events. That would be an example of event that has a timespan and is not punctual."
- When scroll disappears, timeline size is off:
  - "Splitting a timeline sometimes create a blank space at the end of the timeline."

3.2.4 Magnets Grid

- Shaking magnets:
  - "I would like the shake magnets function to be called when I drag the magnets. If we don't want it to go too fast, it can also happen only when we drop them to new positions."
  - "Shaking a magnet does nothing we have to click on the small button. Shaking should happen when we click on the magnet."
- More tools for dynamically creating magnets/attributes

3.2.5 Map and Timeline

- "It would be nice to be able to launch the empty map from the workspace tab, or have it already there in the workspace when we begin."
  - Can save Workspace state
- "There is no fill color in the route ribbons, would that be difficult?"
  - Decided against to prevent clutter
- "Route ribbons should not appear for vessels where it is meaningless, although I admit it would be difficult to select which ones should have it. For some tracks, it goes over land."
  - Design choice

3.2.6 Pop-ups

- Confirmation on delete of items

3.2.7 Workspace

- SmartGWT breaks Firefox:
  - Improvements have been made to Firefox and IE9 support but issues remain

3.2.8 Graph Analyzer

- "It would be nice to be able to collapse and remove nodes in the graph"
  - Other MVAP widgets operate on the idea of "select and export" to filter down the size of a set. We wanted to keep this consistent in the Graph Analyzer.

4 Widget Application Shell (WAS) Development

During Phase 1 of the Maritime Visual Analytics Prototype Development project, an amendment was made to the contract for Architecture and Configurable Workspace Development to improve the underlying web development framework known as the Widget Application Shell (WAS).

Oculus responded to the framework amendment by analyzing and improving upon the WAS. Significant additional features implemented by Oculus include the ability to save and restore workspace state, integration of the open source authentication and access management platform OpenAM, support for multiple instances of the same type of widget and support for installation specific configuration files to control both client and server behaviour.

Oculus identified WAS limitations and performed an evaluation of a competing framework, the Ozone Web Framework (OWF) as well as its complementary Synapse Framework. For the purpose of addressing the Architecture and Configurable Workspace Development functional and non-functional requirements, it was determined to be beyond the scope of the current contract to replace the WAS framework. However, a survey is provided for the OWF and Synapse libraries as well as an analysis of the feasibility of moving existing work to the Ozone Web Framework should WAS be deemed unsuitable in the longer term.

This report section outlines the performed development work on the WAS framework, the evaluation of the competing Ozone Web Framework, analysis of the state of the WAS framework as well as a developer guide to leverage the new support implemented by Oculus.

4.1 WAS Framework: Context, Objectives and Scope

4.1.1 Context

Under the Maritime Visual Analytics Prototype Development contract, Oculus has completed work on prototyping a selected set of widgets and services to provide maritime visual analytics capabilities. These components have been integrated into the I2 section's Widget Application Shell (WAS) framework.

In parallel with this effort, Oculus has undertaken architecture and prototyping of a configurable workspace as well as other enhancements over capabilities currently implemented in the existing VOiiLA framework.

At present there are a number of initiatives underway in government, the private sector and open source communities to develop "container" frameworks for hosting and orchestrating visual widgets within a browser. Oculus has conducted a survey of current competitive third-party frameworks to assess alignment with high level DRDC goals, compliance with specified DRDC requirements, compatibility with I&I architecture principles, frameworks and infrastructure, and the estimated effort of integration and any required adaptation.

4.1.2 Objectives

The objectives of this contract amendment are:

- Provide enhanced configurable workspace capabilities in compliance with DRDC specified architectural, functional and non-functional requirements.
- Develop prototype software implementation for integration into I2 section infrastructure.
- Capture knowledge from the development process in support of subsequent projects.

4.1.3 Deliverables

The deliverables for this statement of work include:

- This report detailing the software and framework research effort.
- Software deliverables for the prototype development.
- A developer usage guide for prototype code as well as guidance for future work.

4.1.4 Framework Alternatives

Oculus has identified WAS limitations and performed an evaluation of a competing framework, the Ozone Web Framework (OWF) as well as its complementary Synapse Framework.

For the purpose of addressing the Architecture and Configurable Workspace Development functional and non-functional requirements, it was determined to be beyond the scope of the current contract to replace the WAS framework.

However, a survey is provided for the OWF and Synapse libraries as well as an analysis of the feasibility of moving existing work to the Ozone Web Framework should WAS be deemed unsuitable in the longer term.

4.2 Oculus Development Work

Below is a high-level summary of the WAS framework development work. Where necessary, additional details and code samples have been provided in the WAS Developer Guide section.

4.2.1 Secure User Login

The OpenAM platform (http://forgerock.com/openam.html) is an access management, entitlements and federation server platform. The OpenAM suite provides a comprehensive solution for authentication of users and authorization of services across multiple applications.

OpenAM is an open-source project managed by ForgeRock and implemented 100% in Java. This allows for easy and fast deployment in a variety of configurations.

Oculus has deployed a fully-functional OpenAM distribution on its test servers. Users attempting to access any WAS web application will be forced to login to the deployed OpenAM server for

authentication. Successful authentication will result in automatic redirection to the desired application.

4.2.2 Client/Server WAS Settings & Configurable Workspaces

Oculus has architected a system for easy configuration of any WAS framework-based application. This system requires no rebuild of the underlying source when changes occur and is easily managed with any off the shelf text editor.

Under the current prototype configuration, Oculus has selected Apache Tomcat (http://tomcat.apache.org/) for web server deployment. As an established open source solution implemented entirely in Java, Tomcat balances ease of deployment along with proven reliability.

When deploying any WAS-based application, the user may optionally provide two plain text files into the */conf* directory of the Tomcat installation. They must be named *was_client.properties* and *was_server.properties*.

Settings in the client file are automatically ingested by the system and efficiently placed in the global Javascript via JSP so that all settings are available at application startup. Developers can access these settings programmatically via the `WasPropertyManager` object found in their `WASEntryPoint`. In particular, client settings can be used to create workspace specific settings.

Settings in the server file are ingested and placed server-side in the `ServerPropertyManager`. Sample settings: database location, installation URIs, etc.

4.2.3 Persistent Workspaces

The ability to save and continue work between sessions is paramount for application usability. Oculus has architected a system for saving user workspaces and widgets and implemented it in the WAS framework. This system has been used by the Maritime Visual Analytics Prototype Development prototype. For the prototype implementation, Oculus has opted for the open-source database PostgreSQL (http://www.postgresql.org/).

In our proposed API, widgets and workspaces must implement a method for exporting the widget state. A service, `PersistenceService`, manages the automatic saving of any and all workspaces in the user's current session. Saving the contained widgets is automatically committed to the database by our service.

When a WAS application is started, a service call is made to retrieve the last state for a user. On successful retrieval, widgets and workspaces are reconstructed from their saved states.

4.3 WAS Limitations

While the preceding work addresses much in the way of usability for WAS framework-based application, there remain a few key issues we outline below.

4.3.1 Google Web Toolkit (GWT) Limitations

The Google Web Toolkit is an open-source suite of tools allowing developers to create and maintain JavaScript front-end applications in Java. This allows traditional Java developers to leverage existing expertise, Java libraries and to more quickly develop web applications without having to learn JavaScript. As a higher-level language, Java is better suited towards creating structured extensible code than JavaScript, traditionally oriented towards scripting tasks.

The GWT is not without numerous drawbacks that affect the future of the WAS framework. Unlike pure JavaScript, the GWT compiler must build JavaScript files each time the Java source is changed. This is often an intensive time-consuming compilation before any debugging can take place. This is a significant hindrance to developer productivity.

GWT's cross-compiler, while powerful, has limitations. The compiler can produce unexpected behaviour. For example, simple Java code might be converted to very bulky and inefficient JavaScript code. In many cases, the loss of clarity and performance makes a straightforward implementation in JavaScript preferable.

4.3.2 SmartGWT Limitations

SmartGWT (http://code.google.com/p/smartgwt/) is a GWT-based framework which provides additional widgets and server-side integration functionality. It is based upon the SmartClient Ajax framework (http://www.smartclient.com/) and provides access to this comprehensive open source widget collection.

There are many drawbacks to using SmartGWT. Most significantly, the open-source LGL licensed version of SmartGWT offers no developer support and contains several glitches that affect the WAS framework. For example, all windows objects in Firefox have a 100 by 100 fixed view size which is difficult to work around.

Another notable issue is that creating a single SmartGWT element will often lead to DOM elements nested 3 or 4 levels deeper than necessary. This is highly inefficient and can lead to issues when the WAS framework needs to access said elements. The additional level of indirection introduced by SmartGWT exacerbates the performance problems with GWT mentioned above.

4.3.3 WAS Issues

There remain architecture issues with the WAS framework that should be addressed going forward.

4.3.3.1 Tightly Coupled Compilation

An ideal framework for hosting widgets would allow for each widget to be developed independently of the framework as well as other widgets. In addition, a clean communication interface should be provided for data transfer between widgets.

The current state of WAS requires that any application be aware of its supported widgets during the build process. As such, widget developers need to rebuild the WAS framework application to

deploy new widgets. This would lead to inefficient and wasted effort if multiple teams were tasked with building and testing widgets on the same application instance.

4.3.3.2 User Stories and Workflow Design

Most successful application development involves user-focused design. Development methodologies such as Agile focus on "User Stories" to analyse the workflow of the end user and create applications which simplify the workflow as much as possible.

In an open framework such as WAS, the user has the flexibility to open many arbitrary widgets. While this flexibility seems appealing, the danger is that the end-user workflow is not clearly defined. It is not obvious to the user how or when to transition from one tool to the next.

Oculus has extensive experience deploying highly interactive applications and has learned that applications with a clearly defined workflow are more readily adopted by end users.

When developing applications in WAS, the transition between widgets should be made clear with obvious controls. Expecting the user to have many widgets open simultaneously or to access a list of available widgets as a step in a standard workflow are likely to produce a poor user experience.

WAS is designed to be one web application that exposes a large range of functionality through workspaces and widgets. An alternative would be to create distinct task-focused web applications. In practice, task-focused applications are superior for simplifying user workflow.

4.3.3.3 Facilitating Widget UI Layout

The WAS framework does not provide an adequate system for controlling the size and layout of widget components within a WAS window. There should be a simple mechanism for informing the widget of the size and position of the window so that it can update internal UI components appropriately.

4.4 WAS Alternative: Ozone Widget Framework and Synapse

4.4.1 Ozone Widget Framework (OWF)

Ozone is a US government open-source software solution for displaying lightweight browser widgets in a single application. Ozone has two primary roles: layout manager and a messaging mechanism.

First, Ozone is responsible for hosting arbitrary widgets by placing them in *iframes*. Once a widget is added, Ozone is responsible for providing various different layouts (e.g. tabbed, portal, etc) allowing the user to see a unified interface. This allows for the creation of consistent dashboard-like interfaces out of widgets from disparate sources.

By using iframes to host widgets, Ozone makes it possible to turn almost any web application into an Ozone widget. A widget's iframe window displays what would otherwise be the entire contents of the web page at a given address. This mechanism allows for independent development of widgets which can later be integrated into the framework with minimal effort.

As JavaScript does not allow communication between *iframes* from different domains, Ozone provide a cross-domain-compatible publish-subscribe mechanism for inter-widget communication. It is by this mechanism that several widgets from different domains can communicate mission critical information.

OWF adoption rate has been very high due to the popularity of the widget-based architecture as well as the *Ozone Marketplace*, a central hub for distributing and obtaining widgets.

4.4.2 Synapse

The Synapse platform is an open-source SDK aimed at augmenting the Ozone development platform by introducing additional data and communication standards. The key contributions are: an improved data model, a shared storage mechanism and an enhanced communication model.

The *Synapse Common Data Model* provides a data representation to be used by all widgets. This record-based representation aims to provide predictable underlying structure to data for data administration while still allowing expressiveness of structure for widget developers. There is an API provided to map existing data for ingestion into the Synapse system. The data model encourages developers and software architects to model their data in a manner that will be familiar to all developers working in the Synapse framework.

Shared storage is achieved by creating a special type of widget termed the *source widget*. These widgets act as data gatekeepers. They are responsible for data ingestion and instantiating the *common data model* to be used. Visual widgets needing access to the data will communicate through this widget. Architecturally, this widget plays a *model* role in the standard "model view controller" paradigm. Leveraging these *source widgets*, developers can easily check if the data is available and initialized for use. This greatly saves on both space and time performance by avoiding unnecessary duplication of resources.

The *Synapse Common Eventing Model* (CEM) aims to create a standard set of events and communication channels which Ozone does not outline. Common events such as record deletion or addition are defined and appropriate OWF channels are used to further modularize the communication model. As always, APIs allow easy subscription and publishing functionality for widget developers.

4.5 Future Work

4.5.1 Further WAS Improvements

The following improvements are suggested to increase the usability of the framework.

4.5.1.1 User Sharing

In web based applications data is stored on a central server and could be shared amongst distributed users. With the addition of persistent workspaces, a possibility exists for users to share workspaces or widget state with other users. In addition, mechanisms could be added for live chat between users, commenting and task assignment. There is currently no functionality provided by the framework for this sort of user interaction.

4.5.1.2 Data Model Architecture Design

The initial WAS prototype lacked the presence of a data model, particularly at the *workspace* level. Oculus has taken steps to address this issue and has learned the key features required for developers wanting to create their own *workspaces*. These include, but are not limited to: registering widget support, data exporting functionality and widget re-creation on application start up.

It remains to extend the framework data model and usage pattern to allow for a system to dynamically add and remove workspaces.

A system could be developed to template workspaces so that users could quickly initiate a task in a new workspace with appropriate widgets and default data. This template could then be modified by adding or removing widgets, and changing data sources or filters.

Workspaces templates and instances could be shared amongst users to create common operating procedures.

4.5.1.3 Workspace Administration

If WAS is to be deployed as a large multi-user system, there will need to be administrative users and tools to support the end users. For example, if new widgets become available, it would be useful to allow an administrative user to push the widgets into the end user workspaces or at least to notify the users of the availability of new tools. Similarly, administration of common data will become necessary either to control access or to manage backups and elimination of outdated data.

4.5.1.4 WAS Replacement

Should the WAS prototype be deemed unfit for further development, it is our opinion the Ozone/Synapse combination aligns best with the high-level goals of the DRDC. Ozone/Synapse provides a proven open-source software solution leveraging the best aspects of JavaScript but also providing structured communication and data modeling.

Should the Ozone/Synapse framework be adopted, much of the current WAS effort could be adapted with relative ease:

- Existing widgets: Ozone supports any JavaScript in an *iFrame*. Minimum adaptation would be required.
- Saved state: The database effort could be adapted to the new framework with minimal effort.
- Configurable workspaces: The configuration functionality could be moved to a common *controller* widget. Also minimal effort.

The advantages of using OWF and Synapse include:

- Proven open-source solution with US government backing
- Established user base and associated community expertise
- Well-architected data model and communication solutions
- Flexible widget hosting of any web application

- Properly decoupled development environment, great increasing productivity

While these gains are significant, we feel it is necessary to identify the risk of adopting foreign government controlled software – even one that is open-source.

4.6 WAS Developer Guide

4.6.1 Client WAS Settings & Configurable Workspaces

Client settings are automatically read from the file "was_client.properties" in the *{tomcat_install}/conf* directoryon the server. Settings are presented as key-value pairs in the plaintext file. For example, "`your_client_setting=settingValue`". Administrators will be able to edit this file and changes will be reflected when the main page is next loaded in a browser.

In the source code, WASEntryPoint.java has a method getPropertyManager() which will return a WASPropertyManager object. This object is a thin facade for a hash map containing all key-value pairs in the aforementioned settings file.

4.6.2 Server WAS Settings

Server settings are read from the file "was_server.properties" in the *{tomcat_install}/conf* directory. Settings are presented as key-value pairs in the plaintext file. For example, "`database_host=localhost`". Users will be able to edit this file and changes will be reflected when the server is restarted.

In the source code, an instance of ServerPropertyManager will read in this file automatically. This manager is a thin facade for a hash map containing all key-value pairs in the aforementioned settings file.

4.6.3 Persistent Workspaces

Workspace persistence is implemented using a small flat set of data objects: `UserState`, `WorkspaceState` and `WidgetState`. The act of saving and fetching the user's data is done by the `PersistenceService`, a GWT service.

Persistence Objects:

- `UserState`: Used to store all workspaces for a single user
  - `owner`: The user's unique id/name
  - `workspaces`: A list of `WorkspaceState` objects corresponding to this user's workspaces
  - `settings`: A set of key-value pairs for storage of persistent settings

- `WorkspaceState`: Used to store a single workspace and its widgets
  - `widgets`: A list of `WidgetState` objects corresponding to this user's widgets

    - `settings`: A set of key-value pairs for storage of persistent settings

- `WidgetState`: Used to store a single widget
    - `settings`: A set of key-value pairs for storage of persistent settings

The PersistenceService provides two methods:

- `getStateForUser(String id)` – Fetches the `UserState` object for the desired user
- `saveUserState(UserState state)` – Commits the `UserState` to the database

4.6.4 Tutorial: Saving Data

For the purposes of our example consider a custom workspace and widget type to which we wish to add the ability to save to the database. We consider the case of a weather-based workspace which has a global setting for the type of temperature units – Celsius or Fahrenheit. This workspace will support one widget whose purpose is to display the weather around an airport. It will have a setting for its corresponding airport.

First, we create the workspace extending the base class `WASWorkspace` called `MeteorologyWorkspace`. We override the `exportStateMethod()` and add boilerplate code for the temperature unit type:

```
public class MeteorologyWorkspace extends WASWorkspace {

	private Boolean useCelsius;

	public MeteorologyWorkspace(WASApplication pApplication) {
		super(pApplication);
		useCelsius = true;
		// Init any additional settings
	}

	@Override
	protected void initWidgets() {
		// Initialize widgets for this workspace
	}

	@Override
	protected String getName() {
		return "Meteorology Workspace";
	}

	public Boolean getUseCelsius() {
		return useCelsius;
	}

	public void setUseCelsius(Boolean useCelsius) {
		this.useCelsius = useCelsius;
	}

	@Override
```

```
	public WorkspaceState exportState() {
		WorkspaceState state = super.exportState();
		// TODO save information relevant to this workspace
		return state;
	}
   }
```

Next, we ensure that our decision of whether or not to use Celsius degrees is preserved. We add a static key for our new setting and add it to the outgoing workspace state:

```
public class MeteorologyWorkspace extends WASWorkspace {

	public static String USE_CELSIUS =
"MeteorologyWorkspace.use.celsius";

	...

	@Override
	public WorkspaceState exportState() {
		WorkspaceState state = super.exportState();
		state.addSetting(USE_CELSIUS, useCelsius.toString());
		return state;
	}
}
```

Now the persistence service will be able to save this object properly. Let us do the same for the widget:

```
public class AirportWeatherWidget extends BaseWidget {
    public static String AIRPORT_CODE =
    "AirportWeatherWidget.airport.code";

	private String airportCode;

	public AirportWeatherWidget(String airport) {
		super();
		airportCode = airport;
	}

	public String getAirportCode() {
		return airportCode;
	}

	public void setAirportCode(String airportCode) {
		this.airportCode = airportCode;
	}

	@Override
	public WidgetState exportState() {
		WidgetState state = super.exportState();
		state.addSetting(AIRPORT_CODE, airportCode);
		return state;
	}
}
```

Note that the static keys used could be placed in a central location, as is the case for `PersistentProperties` used by the WAS framework.

With these simple changes, if a `MeteorologyWorkspace` is found in the `WASEntryPoint` during a save operation, it will automatically be converted to these state objects and committed to the database via the service call.

Should one wish to use a custom GWT entry point, we need only add the following callback to our code:

```
private PersistenceServiceAsync persistenceSvc =
      GWT.create(PersistenceService.class);

public void doSaveUserState(UserState state) {
AsyncCallback<Boolean> callback = new AsyncCallback<Boolean>() {

      @Override
      public void onFailure(Throwable caught) {
            // handle failed save
      }
      @Override
      public void onSuccess(Boolean result) {
            // handle successful save
      }
};

  persistenceSvc.saveUserState(state, callback);

}
```

4.6.5 Tutorial: Loading and Re-creating Objects

Retrieving the save state is similar to committing the save state. We need only make a call to the PersistenceService service and pass it a user ID. We can use the following snippet:

```
      public void getSaveStateForUser(String userID) {

            AsyncCallback<UserState> callback = new
        AsyncCallback<UserState>() {

                  @Override
                  public void onFailure(Throwable caught) {
                        // no saved state found, load defaults.
                  }

                  @Override
                  public void onSuccess(UserState result) {
                        // state successfully retrieved. load it.
                  }
            };

            persistenceSvc.getStateForUser(userID, callback);
      }
```

To recreate the actual Java objects, we recommend the Factory pattern to best modularize the creation of the widgets based on type. The core functionality for workspaces would be as follows:

```
public static WASWorkspace createWorkspaceFrom(WorkspaceState
            workspaceState, WASApplication app) {
      WASWorkspace ws = null;
      String type =
      workspaceState.getSetting(PersistentProperties.WORKSPACE_TYPE
      );

      if (type == null) {
            // no type. log error.
            return null;
      }

      if (type.equals(MeteorologyWorkspace.TYPE)) {
            String useCelsius = workspaceState.getSetting(
               MeteorologyWorkspace.USE_CELSIUS);
            ws = new MeteorologyWorkspace(useCelsius);
      }

      // boilerplate for other workspaces
      /**
      if (type.equals(OtherWorkspace.TYPE)) {
            ws = new OtherWorkspace();
      }
      **/

      for (WidgetState widget: workspaceState.getWidgets()) {
            // add widgets. Use:
            ws.setDeferredWidgets(workspaceState.getWidgets();
            // explanation to follow
      }

      return ws;
}
```

Similarly for widgets we would have:

```
public static BaseWidget createWidgetFrom(MVAPWorkspace workspace,
WidgetState ws) {
      String type =
ws.getSetting(PersistentProperties.WIDGET_TYPE);
      if (type == null) {
            // no type, log error
            return null;
      }

      if (type.equals(AirportWeatherWidget.TYPE)) {
            String airportCode =
                  ws.getSetting(AirportWeatherWidget.AIRPORT_CODE);
            return new AirportWeatherWidget(airportCode);
      }

      return null;
}
```

We can now re-create the associated widget and workspace objects but one detail remains: why use this *setDeferredWidgets()* method? Due to a WAS limitation, widgets can only be created once its parent workspace has been created and rendered visually.

We avoid this limitation by delaying the creation of widgets until after we are certain the parent workspace has been rendered. Rather than immediately creating the widgets after the workspaces, we pass the widget states to the parent workspace as *deferred widgets*:

```
public void setDeferredWidgets(ArrayList<WidgetState> toDefer) {
      deferredWidgets.addAll(toDefer);
}
```

Once this is done, we simply create the widgets only after we are certain the parent workspace has been rendered. We recommend doing so in the method *initWidgets()* on `WasWorkspace`.

4.7 OpenAM Deployment

The following is a guide to installing the open-source authentication system OpenAM on an Apache Tomcat server. We will also show how to protect a web application. For Tomcat shell scripts, we use the Linux nomenclature but all scripts have windows equivalents in the OpenAM distribution.

Caveat 1: Due to a compatibility issue with Tomcat 7, there is no J2EE client agent for OpenAM protection yet. Thus we will use Tomcat 6 for our deployments. To our knowledge, all other steps are the same for other web servers.

Caveat 2: Even for testing, OpenAM should not use the local domain *localhost*. OpenAM requires a fully qualified domain name to function properly.

Caveat 3: The OpenAM and web application should be in different Tomcat installations.

We assume the following have been downloaded:

- Java 1.6 (also installed): http://java.com/en/download/index.jsp
- openAM: http://download.forgerock.org/downloads/openam/snapshot9.5/openam_954.war
- Tomcat web agent: http://download.forgerock.org/downloads/openam/j2eeagents/stable/3.0.3/tomcat_v6_agent_303.zip

4.7.1 Server Pre-deployment Configuration

OpenAM is somewhat memory intensive and requires the following arguments to increase the JVM memory allotment:

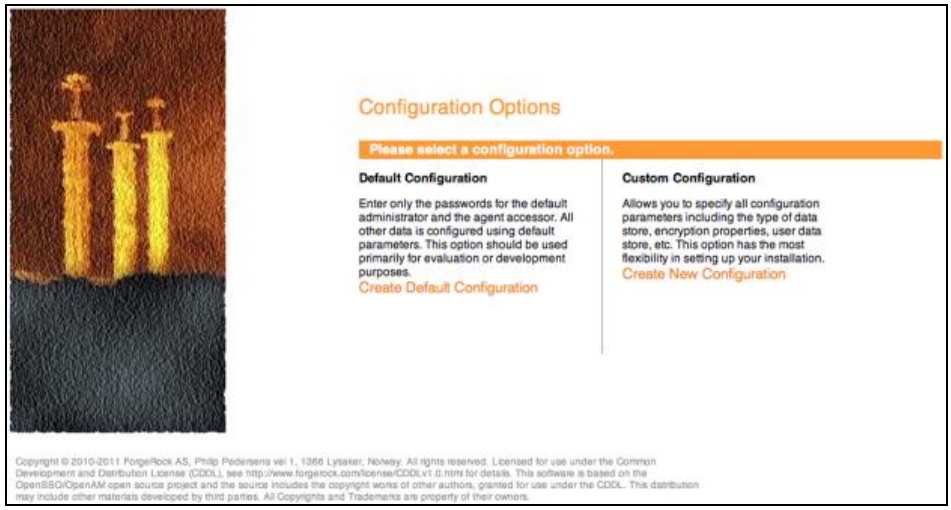
```
-Xmx1024m -XX:MaxPermSize=256m
```

Other relevant considerations at this point might include port selection, SSL vs non-SSL access, etc.

4.7.2 Server Installation

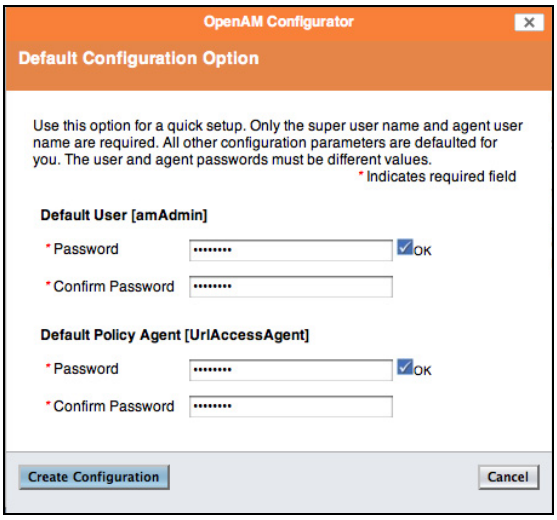
To install openAM, we need only drop the *openam_954.war* file into the {*openam_tomcat_directory*}*/webapps* directory. Once copied, one can start the server via {*openam_tomcat_directory*}/bin/startup.sh

If we navigate to {*openam_tomcat_url*}/opensso we will see the following:



4.7.3 Server Configuration

While OpenAM allows for command-line configuration, we opt for the graphical configuration as seen above. Proceed with "Default Configuration".



At the above screen, choose your admin password and the default policy password. Note these should be distinct. The admin password is the one used to access and administrate OpenAM. The policy password is the fallback password for external agents to interact with OpenAM.

For the “Server Settings” step - so long as the URL being accessed is not localhost, the auto-detected entries will be correct.

For “Configuration Store”, the built-in OpenDS will suffice for most prototype purposes. Fielded installations will need to have a separate OpenDS installation or other data store for deployment.

For the remaining tabs – “Site Configuration”, “Agent Information”, we accept the defaults.

OpenAM will summarize the setting before creating the configuration. Once this is complete, we will be redirected to the login screen. This completes the general server configuration.

4.7.4 Protecting a Web Application

At this point, we have a functioning installation of OpenAM. Assume we have a web application we wish to protect via username/password login. Web applications are protected via agents that handle the redirection of traffic and denial of access according to policies in the OpenAM configuration.

The steps for this configuration are:

1. Create users in the OpenAM data store
2. Setting user access permissions via policies
3. Creating the server-side agent component
4. Install the client-side agent component

4.7.4.1.1 Creating Users

Once logged in, click the “Access Control” tab. Under “Realm Name”, we choose the top level domain. Then at the top right we click on the “Subjects” tab, this is the user tab. Notice some users have already been create (e.g. amAdmin, demo, etc).

We can create as many users as we wish now. Simply click on “New...” and fill in the relevant information.

4.7.4.1.2 Access Policies

We now create a policy for accessing our web application. Again, under the “Access Control” tab, we choose our top level domain. Then we click the “Policies” tab. Click “New Policy...” and name this policy “MyWebApp”.

We add a rule (“Rules” > “New...”), choose “URL Policy Agent” and name the rule “MyWebAppRule”. The “Resource Name” will denote the URL being protected which includes the path and its children. We want to put our web application’s parent directory and NOT the server

directory e.g. {*client_url:port*}. Then we check the boxes for "GET" and "POST" and ensure they are set to "allow". We then save this rule.

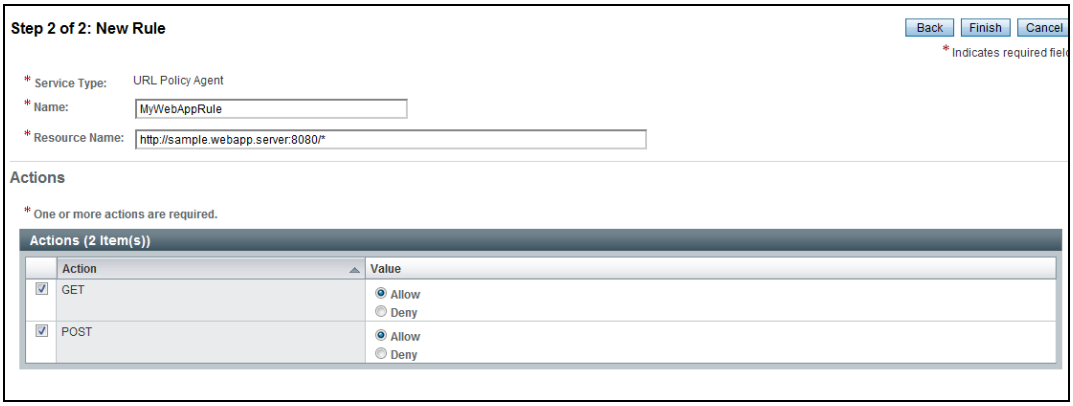


Figure 52: OpenAM: Creating a New Rule

Once back in the policy screen, we want to set the users that are allowed to access this resource. We choose "Subjects" > "New..." and select "OpenAM Identity Subject" to limit access to only those registered with OpenAM. We'll call this group "MyWebAppUsers". Choose "User" from the filter dropdown and click "Search" to see all users. Then add those that will have access to this resource. Save this user set to get back to the policy screen.

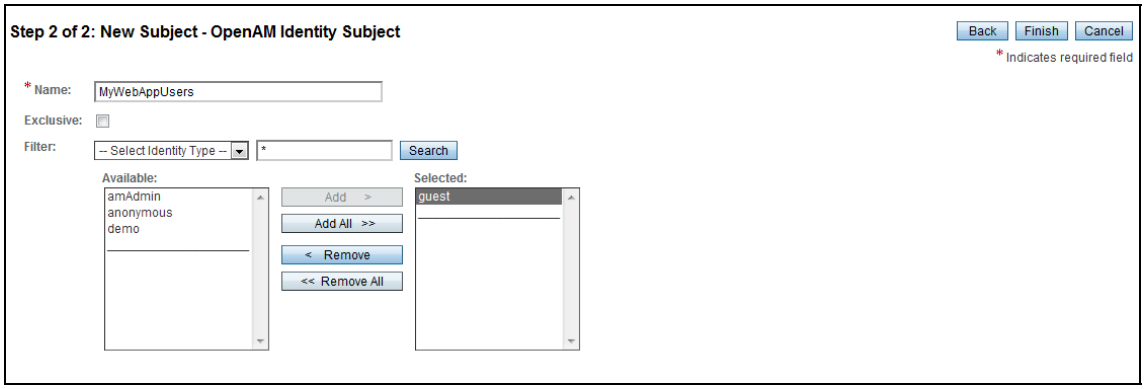


Figure 53: OpenAM: Allowing Access for Users

Click "Save" on the upper right to save this policy. At this point, we have a web policy in place that protects our web application and limits access to only those we added to a list of approved users.

4.7.4.1.3 Creating the Agent Server-Side

Once again, under the "Access Control" tab, we choose our top level domain. This time we click on the "Agents" tab. We want to create a web agent profile for protecting our resource. We need a J2EE agent so we click on that sub-tab. Then we click "New..." under "Agent".

We'll name our new agent "webappagent" and give it an access password. We leave the configuration "centralized" then pass it the running OpenAM URL and the web app deployment

URL. This agent simply tells OpenAM which URL is being protected and by which OpenAM instance. The policy is still what is used to determine the actual access permissions.

4.7.4.1.4 Creating the Agent Client-Side

We can finally move to deploying our web application. As with OpenAM, we move our *application.war* to the {*webapp_tomcat_install*}/webapps directory to deploy it.

We need to create a plaintext password file for the next step. The content should be only the password created above.

Next to install the web agent, we unzip *tomcat_v6_agent_303.zip* to a convenient location – we recommend the base tomcat directory. We then move to {*unzipped_agent_directory*}/bin and run the the following command:

```
% ./agentadmin --install
```

This will begin the command line installation of the web agent for the client. We will enter:

1. The Tomcat server configuration directory for the web application
2. The URL for the OpenAM deployment
3. The Tomcat server install directory
4. The URL for the web application
5. The location of the password file we just created

Once this is complete, the installation program will alter the configurations for the two servers to allow for forwarding when a protected resource is accessed for the first time.

We now start the web application's Tomcat instance. Attempting to access the web application will forward us to the login page on the OpenAM server.

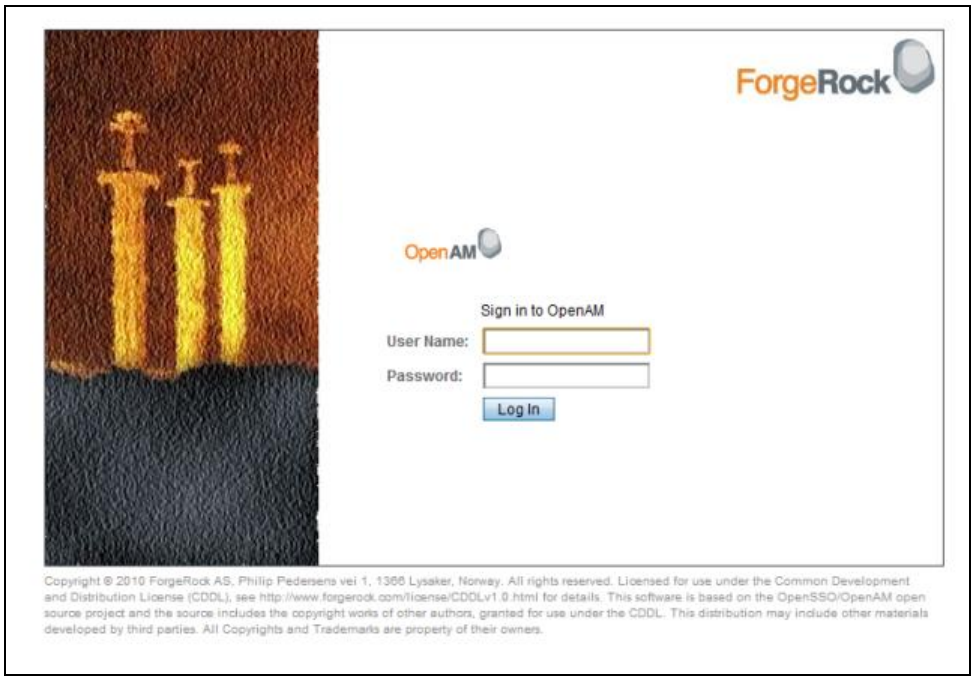


Figure 54: OpenAM: Login Page

5 Demonstration Scenarios

This section describes the demonstration of the Maritime Visual Analytics Prototype using a maritime domain scenario. Section 5.1 describes how maritime domain elements and facts were mapped to the generic elements of the MVAP. Section 5.2 describes an initial scenario that was used to highlight the non-geographic components of the MVAP. Section 5.3 describes the scenario that was demonstrated at the development Phase 1 and 2 project conclusion meeting, and Section 5.4 describes an additional scenario which demonstrates the full features of the MVAP.

Instructions for setting up the demonstrations are given in Section 5.6.

5.1 Data Mapping to the Maritime Domain

The MVAP design described in previous sections of this report is intentionally generic, so that it can be re-used in other domains. This commitment to being generic is illustrated by the database structure in Figure 55, which uses no domain-specific names. Lavigne illustrates the value of this generic approach [10] by describing how Visual Summary Cards and the Record Browser could be used for terrestrial anti-terrorist situation analysis.

For Maritime Domain Awareness, archives represent vessels. For the SNA amendment, in addition to vessels, archives can also represent people or organizations. Links can be between any archive (vessels, persons or organizations). Tracks and events retain their same meaning.

However the demonstration scenarios developed for this project focus on Maritime Domain Awareness, so this section briefly reviews how the generic model was mapped into the domain.

Figure 55 shows how a simplified data model for the maritime domain, taken from an earlier DRDC study, maps to the MVAP services and visualizations. The background diagram in Figure 55 is a simplified view of the Maritime Domain Ontology (MDO) developed to support automated reasoning for maritime anomaly detection (ARMAD). The full MDO model includes far more detail than required for this project, but the upper ontology shown in the figure is relevant.

The key mappings between the MDO and MVAP are as follows, as illustrated in Figure 55:

- **Vessel:** each record in the **archive** table represents one Vessel;
- **Journey:** a journey is a useful data item, because a number of key data elements remain constant during one journey (cargo, crew, originating port, destination, insurance company, etc.). Each record in a **track** table represents one journey. Departure and arrival times for each journey are in the **trackmeta** table. Origin and destination locations are currently stored as event metadata (**eventmeta**) and thus indexed via the vessel/**archive** rather than the journey/**track**. Other journey attributes such as cargo and crew were not used for these demonstrations.
- **Track:** in the MDO, a "track" is just a collection of space-time coordinates, so the **trackpoints** table represents a "track" in the MDO.
- **Dossier:** the MDO postulates an open-ended collection of information for each vessel including static (tombstone) information and time-linked (event) information. Tombstone

information (facts about the vessel) is stored in the **archivemeta** table. Vessel of Interest (VOI) designation is indicated in the **archivemeta** table. Time-linked dossier information such as call-ins, arrivals, departures, destinations, etc. are stored in the **event** table.

The MVAP applications each work on one or two of these data tables, as shown in Figure 56.

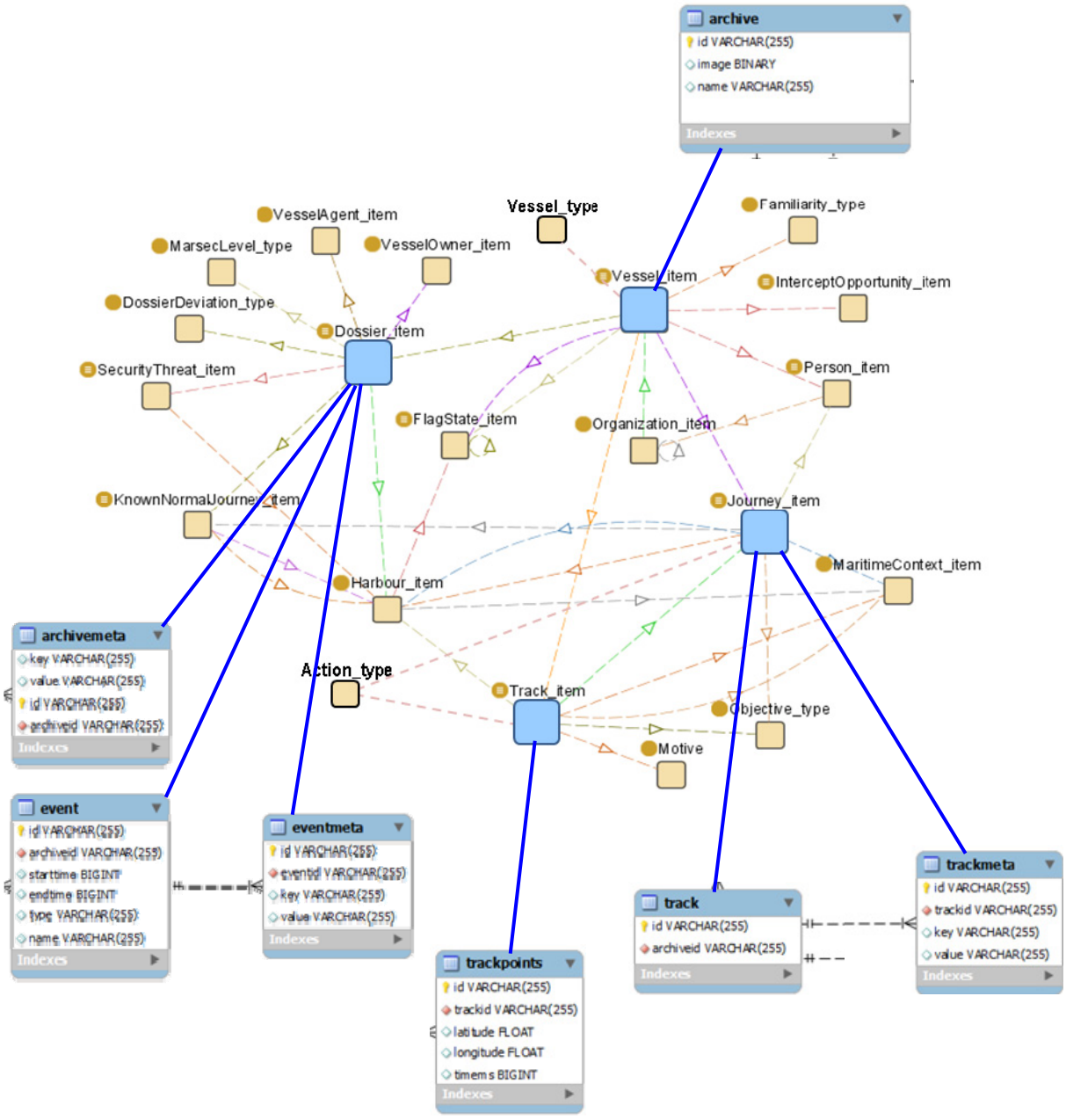


Figure 55: Key Elements of Maritime Domain Situation Assessment

Vessels that go on journeys are key elements of the maritime domain. Each journey has a departure point and a destination, has events, carries a fixed group of people, has an objective (e.g. catch fish, transport cargo), normally

involves only one cargo, and follows one track. A vessel has a dossier that describes past events, journeys, security issues, and commercial information. (Figure is from Davenport, ARMAD slides, Sept 2009). The corresponding MVAP elements, shown as blue boxes, refer to Figure 1.

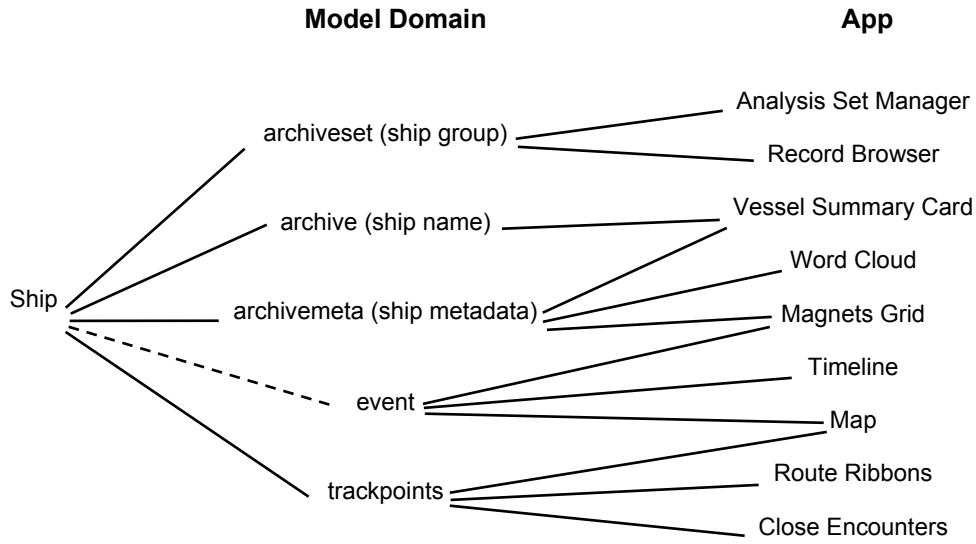


Figure 56: Primary Sources of Data for the MVAP Apps
For the Maritime Domain demonstration, the eight MVAP apps have the names shown on the right. Most apps are predominantly concerned with one or two data sources, as indicated by the connecting lines.

5.2 IRISL Scenario

This section describes the scenario and datasets used for the Phase 1 demonstration of the Maritime Visual Analytics Prototype. This scenario is based on DRDC's "Vessels of Interest" vignette [9], in which Canada watches carefully for ships owned by the Islamic Republic of Iran Shipping Lines (IRISL). Some readers will recognize that many elements of this analysis are normally done by CBSA, who would then pass on the identities of the VOIs to the other MSOC agencies.

Note that, because this document is unclassified, it may be necessary to fictionalize the following words and acronyms to avoid offending other government departments:

Word Used	Replacement
Iran	Jacardia
Islamic Republic of Iran Shipping Lines	Incodia Ocean Shipping Lines
IRISL	IOSL

5.2.1 Background: IRISL

The United Nations has imposed sanctions on Iran for ignoring successive UN Security Council resolutions to cooperate fully with the International Atomic Energy Agency. Canada and its allies have agreed to avoid providing a vessel owned or controlled by, or operating on behalf of the Islamic Republic of Iran Shipping Lines (IRISL) with services for the vessel's operation or maintenance. IRISL has responded to these sanctions by changing the names of its vessels and operating through affiliate companies.

The scenario uses the following definitions, which are derived from [9]:

- IRISL Individual of Interest (IIOI): person known to have previously acted on behalf of IRISL;
- IRISL Controlled Company (ICC): company owned or controlled by IRISL or affiliated entities;
- IRISL Blocked Vessel (IBV): vessel owned by IRISL or an ICC, or previously owned by IRISL or an ICC and not yet "unblocked" by intelligence agencies;
- IRISL Company of Interest (ICI): company meeting any of the following criteria: a) is an ICC, or b) is owned or managed by an IIOI, or c) owns or operates an IBV, or d) has too many connections (this is a fuzzy criterion) to IRISL.
- IRISL Vessel of Interest (IVOI): a ship meeting either of the following criteria: a) is an IBV or b) is owned or operated by an ICI.
- IRISL Human Name List (IHNL): list of person names (first names, last names, aliases) associated with IRISL.
- IRISL Place Name List (IPNL): list of place names (street names, country names, building names) associated with IRISL offices.

5.2.2 Description of the Scenario

This section walks through the scenario to show current functionality of the prototype.

5.2.2.1 Part A: Loitering Vessels

- **Step 1:** The operator selects the ASet 'All Vessels' with approx. 200 ships that are currently within 100 km of Canadian waters and uses the Record Browser to quickly review the cards
  - He flips through quickly, just glancing at each card;
  - 1 of the cards has a VOI flag, and the operator stops briefly to review it, notes that nothing has changed, and moves on.
- **Step 2:** The operator opens the Magnets Grid with the following magnets:
  - Magnet 1 = Loitering <2kts (# of instances of reported speed less than 2knots in 24hrs)

  The operator then sets up a new magnet/attribute called Flag. He sets the Access Name to Flag, and the type to 'Categorical'. He adds the following flag attribute Name/Value pairs:
    - CA - Canada
    - US – United States

- KP – North Korea
- IM – Isle of Man
- GG – Guernsey

In the Dust tab, he sets the Dust Color to Flag.

- **Step 3:** Activating the magnet, four dots stand apart from the others. The operator clicks on each to reveal the ship names. This reveals:
  - Hibernia
  - Maersk Responder - an offshore support vessel tending Hibernia
  - Goreyo Bay - a North Korean fish-packing ship
  - Jaedan & Brothers - a Canadian-owned fishing vessel currently registered in Cypress.
- **Step 4:** The operator then selects those dots, clicks the Export Ships button to create a new Ship Group, and looks at it in the Record Browser. The operator stops at the vessel 'Goreyo Bay' and flips over the Vessel Summary Card to review the vessel dossier. After checking with his supervisor, he clicks a button on the dossier browser to flag this as a VOI. A VOI flag appears on the Vessel Summary Card for that ship.
- **Step 5:** The operator takes a screen shot and copies key information from the Vessel Dossier into a PowerPoint presentation for later presentation to the Admiral, and closes the card browser and Magnets Grid.

5.2.2.2 Part B: Jacardian Vessels

- **Step 6:** The operator then reopens "All Vessels" in the Magnets Grid as follows:
  - Dust tab: Dust Color = flag
  - Dust tab: Dust Size = none
  - Magnet 1 Jacardian People = number of words found in each ship's dossier that are also found in the JHNL;
  - Magnet 2 Jacardian Places = number of words found in each ship's dossier that are also found in the JPNL;
  - Magnet 3 Trips to Canada = total connections trips to Canada, as defined below (this is like the "warm and fuzzy list" [Table 4-11 of CR 2009-295])
  - Magnet 4 Jacardian Ports = average number of visits to Jacardian ports per year.
- The operator interacts with Magnets Grid as follows:
  - All 100 ships are initially near the center of the display
  - He moves Jacardian People to the NW of the working area and taps on it twice, causing about 20 ships to move NW;
  - He moves Jacardian Places to the NE and taps on it twice, cause about 3 ships to move NE. About 3 ships are now starting to separate to the N.
  - He puts Trips to Canada toward the S and taps on it twice. One of the three ships moves down;

- He puts Jacardian Ports on the W side, and taps it to confirm that all three of the identified ships have regular visits to Iranian ports;
- He double-clicks on each of the three ships and notes that none is a VOI, changes the colour axis to: red = VOI, green = non-VOI and notes that the three remaining ships are not VOIs;
- He selects the three ships and clicks on the Export Vessels button to make a new Ship Group called Jacardia.
- He closes the Magnets Grid.

- **Step 7:** The operator explores the new ship group by opening it in the Record Browser. A Word Cloud displays word frequencies for each ship.
  - Words are larger in proportion to how many connections a ship has to that place;
- **Step 8:** The operator views the dossier for each ship and discovers that all are leased to a company called "Quaritch Assets Management."
  - The operator could then use Jigsaw or nSpace [not in the demo] to explore connections between Quaritch and JOSL.
- **Step 9:** The operator opens the Timeline to explore the three ships further. A single timeline appears.
  - The operator notes that "vessel name change" events occur for all three ships at around the same date in 2011.
- **Step 11:** The operator prepares a report for the bi-weekly briefing to the Admiral:
  - He takes a screen capture of the timeline to a PowerPoint slide;
  - He highlights and copies textual information about Quetzel Assets Management to a PowerPoint slide, and uses Google Maps to grab a satellite image of the street address, showing its proximity to IOSL offices on the Isle of Man.
  - He takes a screen capture of the Vessel Summary Cards to add them as images into the PowerPoint slide presentation.

5.3 Development Phase 1 and 2 Project Conclusion Demonstration Scenario

This section describes the scenario and datasets used for the final demonstration of the Maritime Visual Analytics Prototype at the conclusion of Phase 1 and 2 development.

5.3.1 Part A: Launch the demonstration

- **Step 1:** Open a web browser (recommended: Google Chrome)
- **Step 2**: Redirect to http://<MVAP_SERVER>/wasPrototype using a bookmark

5.3.2 Part B: View Vessels and Sets in the Analysis Set Manager

- **Step 3:** Select the MVAPWorkspace from the top-right dropdown menu

- **Step 4:** Open an analysis set in the Analysis Set Manager by clicking the '+'

  Observe the set of vessels listed under the set

- **Step 5:** Select an analysis set and click the Record Browser icon in toolbar

5.3.3 Part C: Scan Vessels in the Record Browser

- **Step 6:** Observe the vessel image, word cloud of recent ports, close encounter popup, track thumbnail and status icons
- **Step 7:** Flip the card by clicking the lower right to view the vessel details
- **Step 8:** Drag the vessel to the left for comparison
- **Step 9:** Click the arrow at the bottom right repeatedly to view the next vessels

5.3.4 Part D: Launch a Map and Fetch Vessels for a Time and Region of Interest

- **Step 10:** In the Analysis Set Manager, make sure no vessels are selected
- **Step 11:** Click on the Map icon in the toolbar
- **Step 12:** Zoom in on the Bay of Fundy using the mouse wheel on the map
- **Step 13:** Zoom in on July 10, 2011 using the mouse wheel on the lower timeline
- **Step 14:** Zoom the upper timeline to make the control bar larger and centered
- **Step 15:** Click the Fetch Vessels icon in the top right

  Observe that vessels with tracks in the given time range and region appear

5.3.5 Part E: Analyze Vessel Motion and Encounters

- **Step 16:** Click on the Layers icon, open the Tracks list and click on 'Carrier'

  Observe that the track for the vessel 'Carrier' is highlighted

  Observe the close encounter popup which has two dark red tracks indicating close encounters within the active time window

- **Step 17:** Click and drag the white area in the control timeline to move the active window backwards until Carrier is at the start of its track (about 9am July 10)
- **Step 18:** Slowly move the time forward until Carrier is passed by 'Gigaloo' on the West side

  Mouse over Gigaloo to see the vessel details

- **Step 19:** Double click Gigaloo to bring up the Vessel Summary Card

  Note that Gigaloo has made trips to Jacardian Ports

- **Step 20:** Slowly move the time slider forward until Gigaloo is intercepted by CarpeDiem
- **Step 21:** Double click CarpeDiem to launch the Vessel Summary Card

Note that Carpe Diem has also made trips to Jacardian Ports

- **Step 22:** Close the Map widget and Layer dialog

5.3.6 Part F: Create an Analysis Set

- **Step 23:** Click on the Export button in the Map

  Choose the "Export All" radio button
- **Step 24:** Click on the "Add Ship Group" button

  Create an Analysis Set called "Fundy ROI"
- **Step 25:** Add the vessels in the ROI to the new set by clicking "OK"

5.3.7 Part G: Analyse the Set in Magnets Grid

- **Step 26:** Select the "Fundy ROI" set in the Analysis Set Manager
- **Step 27:** Click the "Magnets Grid" button
- **Step 28:** Select the Axes Tab and set the X-Axis to Net Tonnage and the Y-Axis to speed

  Observe the scatter plot classifying the vessels over two variables
- **Step 29:** Set the X-Axis and Y-Axis to 'None'
- **Step 30:** Drag the "Jacardian Ports" magnet to the top left
- **Step 31:** Mouse over the magnet and click the activate button 2-3 times

  Observe that three vessels are attracted to the magnet
- **Step 32:** Drag the Speed magnet to the bottom right and activate

  Observe that two vessels attracted to Jacardian ports are also fast moving
- **Step 33:** Drag the Net Tonnage magnet to the top right and activate

  Observe that the Jacardian Ports vessels do not have a large net tonnage
- **Step 34:** Marquee select the three vessels attracted to Jacardian Ports and export to a new analysis set "Fundy VOI" with just the selected vessels

5.3.8 Part H: Mark the vessels as VOIs

- **Step 35:** Select the "Fundy VOI" set in the Analysis Set Manager
- **Step 36:** Launch the Record Browser
- **Step 37:** For each vessel, toggle the VOI button

5.3.9 Part H: Analyse VOIs in the Timeline

- **Step 38:** Select the "Fundy VOI" set in the Analysis Set Manager
- **Step 39:** Launch the timeline

- **Step 40:** Position the red line over June/July 2011 and zoom in
- **Step 41:** Observe that all three vessels had "Name Change" events in June
- **Step 42:** Unlock the timelines and position all three name change events on the red line
- **Step 43:** Zoom in and observe that Carrier arrived in Jarcardia one month prioir to the name change event. CarpeDiem was at sea between Hong Kong and Shanghai. Gigaloo was at sea between Singapore and Los Angeles

5.4 Additional Demonstration Scenario

This section describes an additional domain focused scenario and datasets which could be used to demonstrate the Maritime Visual Analytics Prototype.

5.4.1 Part A: Scan the Cards

- **Step 1:** The operator uses the Analysis Set Manager to open the 'NAtlanticInboundCargo' ASet (trans-Atlantic cargo ships currently in Canada's area of interest) and uses the Record Browser to quickly review the vessel summary cards
  - He flips through quickly, just glancing at each card;
  - One of the cards is an LNG carrier heading from Murmansk to St. John NB and thus has a VOI flag. The operator stops briefly to review it, notes that nothing has changed, and moves on.

5.4.2 Part B: View the Tracks

- **Step 2:** The operator highlights the 'NAtlanticInboundCargo' ASet in the Analysis Set Manager and then clicks on the "map" icon. This brings up a map of the North Atlantic with all ships in the ASet displayed with tails. He slides the time backwards, and sees the ship tracks change accordingly;
  - He slides the time forward past the present into the future. All future tracks are based on declared destinations. He notes that the LNG carrier will pass the entrance to Halifax Harbour just as '*Shado Bulk III'*, a freighter recently from the Mediterranean, arrives there.
  - He takes a snapshot of the track intersection, for future analysis
  - He marks the *Shado Bulk III* as a VOI, for future analysis.

5.4.3 Part C: Route Ribbons Anomaly Detection

- **Step 3:** The operator clicks on "Route Ribbons" and views the tracks.
  - The track of '*Trajekt*' is clearly different from the others because the ribbon is so wide. The track appears to be a rhumb line from Gibraltar to St John Nfld.
  - The operator opens the vessel summary card for *Trajekt* and notes that it is a small old freighter heading from Macedonia to Montreal with a load of containers.
  - He takes a snapshot of the route ribbon, for future analysis

- The operator un-clicks “Route Ribbons”

5.4.4 Part D: Pop-Up Anomaly Detection

- **Step 4:** The operator clicks on “Close Encounters” and views the resulting pop-ups.
  - Several pop-ups show normal patterns for a busy shipping route: ships travelling in the same direction, and passing ships going in the other direction.
  - One pop-up shows something unusual with ship *Tropfer*, in-bound from Hamburg to New York. In the middle of the night fishing boat *Eileen Dover* approaches the *Tropfer*, waits for the larger ship to go by, then backtracks to harbour – indicating a probable package drop-off.
    - The drop-off was not seen in the map view alone, because the Eileen Dover was outbound and is not a cargo ship.
  - The operator takes a snapshot of the route ribbon, for future analysis
  - He marks the *Tropfer* and *Eileen Dover* as VOIs, for future analysis.

5.4.5 Part E: Detailed Analysis of a VOI using Timelines

- **Step 5:** The operator creates an Analysis Set consisting of the *Tropfer* and the *Eileen Dover*.
- **Step 6:** The operator opens a timeline analysis tool for this Analysis Set.
  - He makes 4 copies of the timeline and shifts them to align them all to the arrival of *Tropfer* in New York.
  - Scrolling back in time from that arrival, he notes that in each case, the *Eileen Dover* leaves harbour in the late evening just before the *Tropfer* passes by.
  - The operator takes a snapshot of the aligned timelines, for use in reporting to the Admiral.
- **Step 7:** The operator places the Vessel Summary Cards for the *Tropfer* and the *Eileen Dover* side by side in the Record Browser.
  - He notes that the First Mate of the *Tropfer* has the same surname as the owner of the *Eileen Dover*.

5.4.6 Part F: Magnets Grid Anomaly Detection

- **Step 8:** The operator shifts his attention back to the full NAtlanticInboundCargo analysis set, and opens a Magnets Grid window.
  - He chooses a magnet for “Crew Size”.
  - He chooses a magnet for “Ship Displacement” and places it opposite to the Crew Size magnet.
  - He chooses a magnet for “Year of Construction” and places it at right angles to the first two magnets.
  - He sets dust colour to “Ship Type”

- During interaction with the magnets, he notes that Ship Displacement and Crew Size tend to cancel each other out, because large ships tend to have large crews.
- He also notes that older ships have larger crews than newer ships of the same size, and crew size varies with ship type.

- **Step 9:** The operator notes that one ship is an anomaly, with a much larger crew than is normal for a ship its size.
  - He clicks on the dust for that ship, and notes that the ship is *Trajekt*, which was previously flagged in Step 3 for having an anomalous track.
  - The combined evidence of unusual track and larger-than-normal crew suggests that the ship may be carrying illegal immigrants as false "crew" members.
  - The operator flags the vessel as a VOI.

5.5 Demonstration Videos

A number of videos have been created during the course of development to demonstrate capabilities to scientific personnel at DRDC Valcartier and users in Halifax. These videos have all been uploaded to the DRDC sharepoint site in the folder: VAMDA>Shared Documents>MVAP Dev Phase 3 and Validation Contract>Videos.

- MVAP Standalone Audio – a video with audio demonstrating the core MVAP widgets in standalone mode as per the Validation Exercise training plan.
- MVAPWAS – a video demonstrating all of the components being used within the WAS framework
- MVAP GPW 2013Aug – an annotated video demonstrating the MVAP GPW widget
- GPWEncounters – an annotated video demonstrating the ability to detect encounters for a vessel within a year and a half of GPW data
- GPWLoitering – an annotated video demonstrating the ability to detect loitering vessels within a given time and region using a year and a half of GPW data
- MVAPFilters – an annotated video demonstrating the filtering capability added to widgets in Phase 3

5.6 Demonstration Setup Guide

The setup of the prototype is a four-step process. Steps in 5.6.5 describe how to run the prototype. These steps can be repeated each time the web prototype is run.

5.6.1 Install Apache Tomcat

To run the web application, the first step is to download and install latest version of the Apache Tomcat web server. At time of writing, the latest version of Tomcat is version 7.0.

As Tomcat uses Java and Java Server Pages it requires a minimum Java version, currently Version 1.6 (Version 6). (See http://tomcat.apache.org/whichversion.html, for further details.)

1. If necessary, update Java SE Runtime Environment (JRE) to Version 1.6 (Version 6.0) or later.

   a. Check the current Java version on the installation machine using: www.java.com/en/download/installed.jsp.

   b. If necessary, follow the instructions to update the JRE.

2. Next, download and un-zip Tomcat server files:

   c. Create a new folder under the C:\ drive called 'Apache Tomcat'

   d. Using http://tomcat.apache.org/download-70.cgi, download the binary distributions for the Core '32-bit Windows zip' (or '64-bit Windows zip').

   e. Un-zip the contents to the Apache Tomcat folder.

   f. Ensure that inside Apache Tomcat is a folder entitled \apache-tomcat-7.0.26 (or similar, corresponding to the version number). Within this folder should appear folders \bin, \conf, \lib, \logs, \temp, \webapp, and \work.

3. Next, configure the JRE_HOME Environment Variable:

Tomcat itself is a Java application and does not use environment variables, but the startup scripts use them to prepare the command that starts Tomcat.

   g. Create a new file in the Tomcat \bin directory entitled 'setenv.bat'.

   h. Using a text editor, add the following to the file:

```
@echo off

 set "JRE_HOME=%ProgramFiles%\Java\jre6"

 exit /b 0
```

   Note: If your Java version 6 is located in the "Program Files (x86)" folder use instead the command: set "JRE_HOME=%ProgramFiles(x86)%\Java\jre6".

   i. Save the file and close the text editor.

4. To ensure the installation was successful, start the Tomcat server using steps listed in section 5.6.5.1. Ensure that the server starts; the Tomcat window will display the message "*Info: Server startup in <xxxx> ms*". Shutdown the server following steps in 5.6.5.3.

5.6.2 Install PostgreSQL

PostgreSQL (also Postgres) is the database management system used by the prototype.

5. Download the one-click installer of PostgreSQL:

http://www.enterprisedb.com/products-services-training/pgdownload#windows
(links from: http://www.postgresql.org/download/windows/)

Note: Choose either the 'Win x86-64' or 'Win x86-32' bit version as appropriate.

6. Navigate to the location of the downloaded executable, and run.
   a. Accept the default file location (C:\Program Files\PosgreSQL\9.1)
   b. Accept the default directory to store data (C:\Program Files\PosgreSQL\9.1\data)
   c. For user 'postgres', use password 'admin'
   d. Click 'Next' to install Postgres
   e. Use port # 5432
   f. Accept default locale
   g. At the end of the installation, there is no need to "Launch Stack Builder" when asked.

5.6.3 Download Prototype files & Copy 'war' file into /webapps

7. Create a directory for the MVAP prototype files.
8. From the Sharepoint site, download the MVAP_2012XXXX.zip file to that directory.
9. Unpack the zip file, containing the webapp and database files.
10. Copy or move the WAR file 'wasPrototype.war' directly into the folder: C:\Apache Tomcat\apache-tomcat-7.0.26\webapps
11. Note the location of the database file (mvap2012XXXX.backup) for the next step.

5.6.4 Restore Database

Restore the Database using the PostgreSQL administration tool 'pgAdminIII'.

12. From the Start Menu find and run pgAdminIII

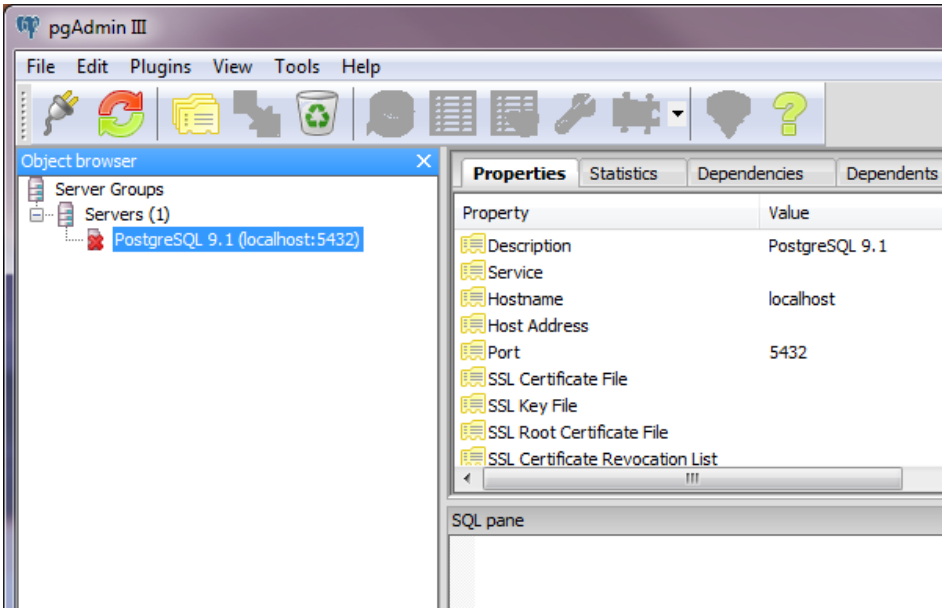


Figure 57: The PostgreSQL Administration Tool

13. To connect the server PostgreSQL 9.1 (localhost:5432), double-click on the server object in the Object browser (left), or right-click and select 'Connect'.

14. Enter the password (admin)

15. Right-click on 'Databases' and select "New Database"

16. Enter name 'mvap' and press OK.

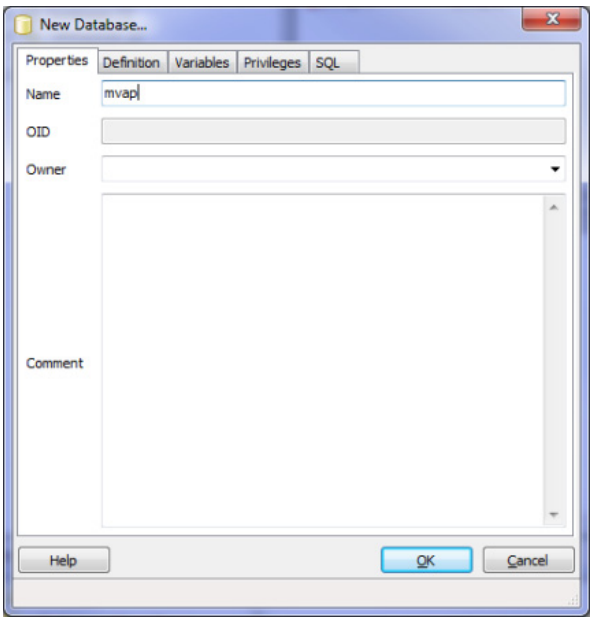


Figure 58: Creating a New Database

17. Right-click on the new mvap Database and select “Restore…”

18. Click the <…> button and navigate to the location of the mvap2012XXXX.backup file. Choose that file and select Open.

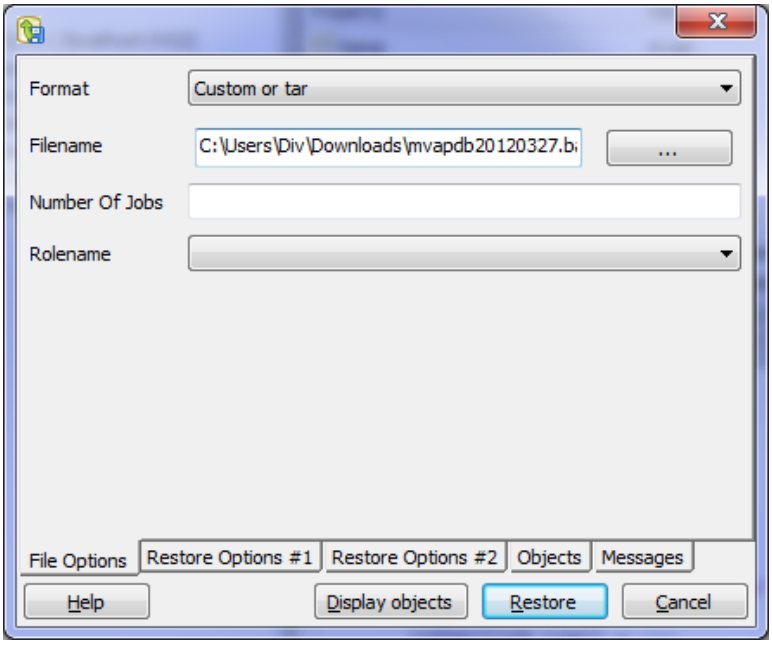


Figure 59: Restoring a Database Backup

19. Choose ‘Restore’, and wait for the restore.

20. Ensure that the Process returns exit code 0 in the Messages pane.

21. *Note that to update a database, the procedure is similar; delete the existing database and follow the steps above to restore the new one.*

22. Close pgAdmin III.

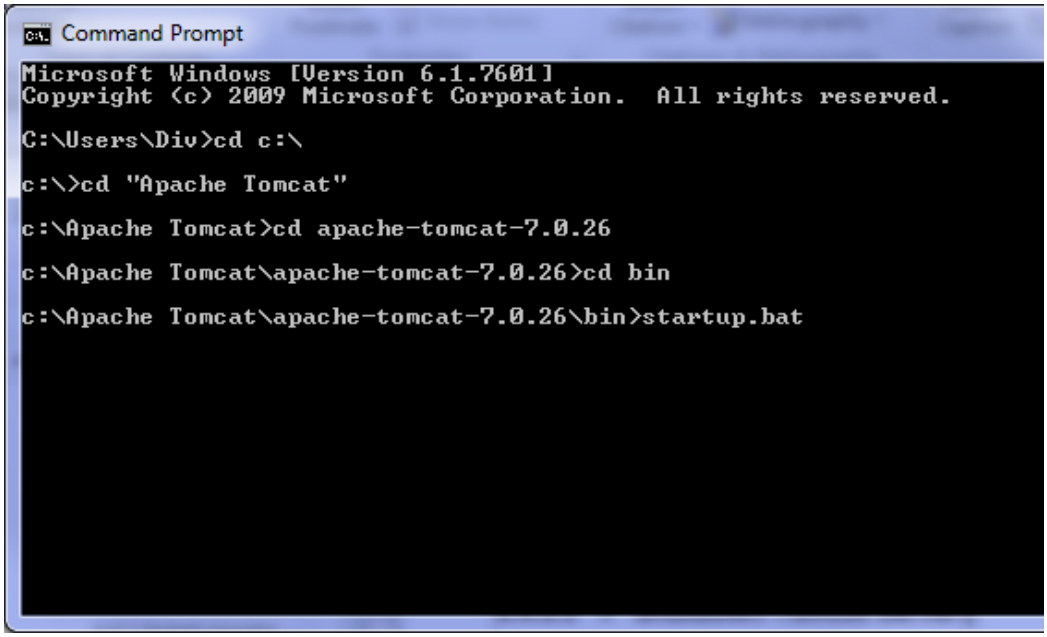
5.6.5 Run the Prototype

To use the prototype, first start the Tomcat server, then open the prototype location in the Google Chrome web browser. *Other browsers may work, but Chrome is recommended. To download Chrome visit: https://www.google.com/chrome.*

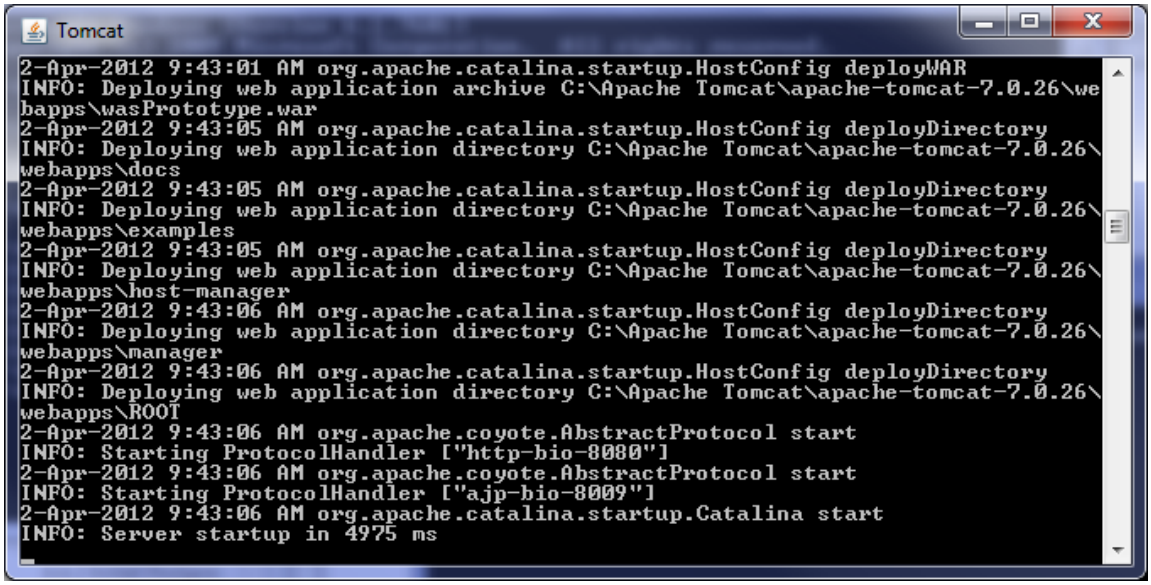
5.6.5.1 Start Tomcat Server

1. In a Command Prompt window (Start | Run: cmd) navigate to C:\Apache Tomcat\apache-tomcat-<version#>\bin

2. Run the command startup.bat



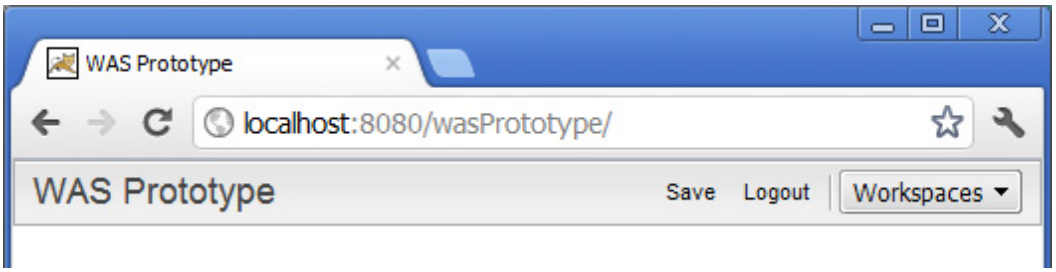
3. This script should open a Tomcat log window (as below)



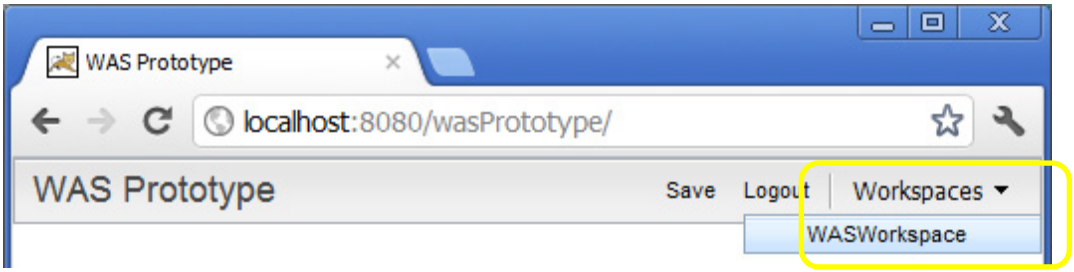
4. The server has started up successfully when the message “INFO: Server startup in XXXX ms” is displayed.

5.6.5.2 Connect to Web Application

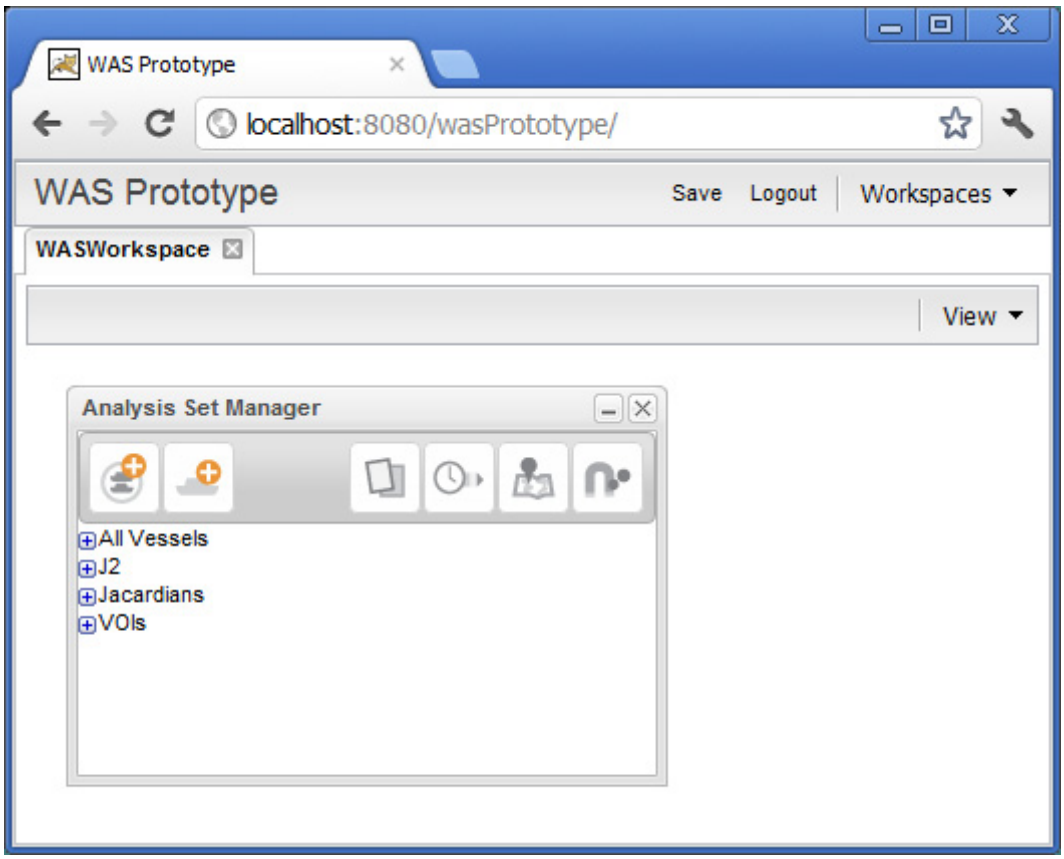
1. Open the Google Chrome web browser, and navigate to: http://localhost:8080/wasPrototype. The prototype page should appear:



2.Select from the 'Workspaces' menu "MVAP Workspaces" (see below).



3. MVAP Workspace should open, within it the Analysis Set Manager should open with several sets already populated (your list of sets may differ from the one below).



4. You are now ready to work with the prototype. See Section 3.1 for a description of prototype functionality.

5.6.5.3 Shutdown Tomcat Server

Only once you are finished running the web application, close the web browser and shut down the Tomcat server.

1. In a Command Prompt window (Start | Run: cmd) navigate to C:\Apache Tomcat\apache-tomcat-<version#>\bin

2. Type in the command shutdown.bat, and press <Enter>.

3. The Tomcat log window should close.

6 Test Procedure

The following test procedure was developed and successfully executed to validate functionality of the MVAP apps. This can also form the basis of regression testing for further development.

6.1 Framework and Workspace

1. Connect to http://<browser_address>/wasPrototype using Chrome or IE9
   - The WAS Prototype framework should appear with a Title, Save, Logout and Workspace headings
2. Select the New... option from the Workspaces menu in the top right
   - Enter a workspace name and select MVAPWorkspace as the type
   - A WASWorkspace tab should appear with a toolbar including a view menu and delete button
   - The Analysis Set Manager should be visible in the workspace
3. Drag the Analysis Set Manager, select save from the top menubar, refresh the page and select the saved workspace from the workspaces menu
   - The workspace should appear with the analysis set manager at the saved location
4. Delete the new workspace using the delete button at the top right
   - The workspace should disappear and be removed from the workspaces menu
5. Create another MVAP Workspace and continue with the test

6.2 Analysis Set Manager

1. The Analysis Set Manager should be visible in the default MVAP workspace
   - The analysis set manager has buttons to:
     - Add a group, Add vessels
     - Launch the record browser, timeline, map and magnets grid
   - The analysis set manager has a tree structure to organise vessel groups
2. Mouse over tree nodes
   - Highlight and display an X to delete the node
3. Clicking on a tree node
   - Selects and displays the X
   - One node can be selected at a time

4. Click on a selected tree node
   - Clears the selection
5. Click on the +/- signs
   - Tree nodes open and close
6. Click the add group button
   - A dialog appears
7. Enter a name and select OK
   - The new analysis set appears in the selected set or the root if no selection
8. Select a set and click add vessels
   - The select vessel dialog appears
9. Click the “+” to add a search criteria
   - A new search criteria line appears
10. Set the new criteria to “Name”, “Contains”, “IRCS” and click search
    - A list of vessels matching the search criteria appears
11. Toggle vessel selection by clicking on vessels and select OK
    - The selected vessels are added to the selected set
12. Click the X beside a vessel or set
    - The vessel or set is deleted

6.3 Record Browser

1. Select an analysis set and click on the launch record browser button
   - The record browser appears
2. Click on the left/right arrows or the scroll bar at the bottom right
   - Different vessel summary cards will appear with:
     - NATO icon
     - Flag
     - 24h/96h indicators
     - Some vessels will have VOI indicators
     - Vessel image
     - Word cloud indicating ports visited

3. Click on the flip button
   - The reverse side of the card appears with a list of vessel properties
4. Click on the gray bar at the bottom of a card
   - It is selected and highlighted (or deselected if already selected)
   - The selection is marked on the scroll bar
   - The selection count is updated in the export button Drag the gray bar to the left empty card space
   - The card appears "locked" on the left side for comparison while changing right cards
   - Flipping or scrolling the left/right cards should be mirrored on the other card
5. Select a number of vessels and click the export button
   - The select analysis set dialog appears
6. Click the Add Analysis Set button
   - Add an analysis set as per the analysis set manager
7. Select an analysis set and select OK
   - The vessels are added to the analysis set and are visible in the analysis set manager

6.4 Timeline

1. Select an analysis set in the manager and click the launch timeline button
   - The timeline widget appears with a timeline for each vessel in the set (max 10)
2. Click the zoom in/zoom out buttons
   - Zooms in/out on the region under the red bar
3. Click on a the red bar, drag and zoom in/out
   - Moves the red bar and changes the zoom center
4. Click on a timeline and drag
   - Pans the timelines
5. Click the lock/unlock button
   - The button is toggled between gray (unlocked) and white (locked)
   - When unlocked, panning a timeline will only pan the individual timeline
6. Click the reset button
   - All timelines are synchronized with the top timeline
7. Click the X button on a timeline

- The timeline is removed from the display (not the set)

8. Click the split button on a timeline
   - The timeline is duplicated
9. Mouse over timelines
   - Shows a light highlight
10. Click on a timeline name or background
    - Selects the timeline allowing for export as per the record browser
11. Double click on an event
    - Finds matching events in other timelines an synchronizes them
12. Click the Zoom to Date button
    - Presents a dialog to select a date to zoom in
13. Click on the filter button
    - Presents a dialog to filter events by time, region and event type
    - New event types can be added with the “+” button
    - Icons can be changed by double clicking
    - Events can be deleted by clicking the “delete” button

6.5 Magnets Grid

1. Select an analysis set in the manager and click the launch magnets grid button
   - The dust and magnets widget appears
   - The vessels are represented as dots in a plot
   - There are tabs on the right to control magnets, dust attributes and plot axes
   - There is a shake button and an export button
2. Click the shake button with no magnets in the display
   - Dust is pushed apart semi-randomly
3. Mouse over dust
   - Vessel information is displayed including parameters affecting active magnets, axes and dust attributes
4. Select dust by clicking, using marquee drag, or holding ctrl
   - Selected dust is highlighted and can be exported as per the record browser
5. Drag magnets onto the plot

- The magnet is grayed out in the tab

6. Mouse over magnets in the plot
   - Play and delete buttons appear
7. Click on magnet play buttons
   - Dust is drawn towards the selected magnet
8. Click on the shake button with magnets in the plot
   - Dust is drawn towards all magnets
9. Click on a magnet X button
   - The magnet is removed from the plot an ungrayed in the tab
10. Click the dust tab to change color and size attributes
11. Click the axes tab to create or remove axes
    - Dust is redistributed according to the axis values
    - Magnet manipulation restricts dust to remain within the axis bins
12. Click on the add magnet button
    - The add magnet dialog appears and allows new magnets to be created
13. Click on the add calculated magnet button
    - The add calculated magnet dialog appears and allows new calculated magnets to appear
14. Click on the filter button
    - The filter dialog appears allowing calculated magnets to only consider events matching the time/region filter

6.6 Map and Timeline

1. Select an analysis set in the manager and click the launch map and timeline button
   - The map and timeline widget appears with tracks for each vessel in the set
2. Use the mouse wheel to zoom in/zoom out on the map
   - Zooms in/out on the mouse cursor location
3. Use the mouse wheel to zoom in/zoom out on each timeline
   - Zooms in/out on the mouse cursor location
4. Click on the top control timeline and drag

- The gray area pans the top timeline, the white area adjusts the window

5. Click on the bottom timeline and drag
   - The timeline window is adjusted on both timelines
   - Vessel positions and tracks are adjusted to reflect the new window
6. Click on the orange time window handles and drag
   - The time window and tracks are updated accordingly
7. Click on a vessel or track
   - The vessel and track are highlighted in purple
   - A Close Encounter pop-up appears if the vessel has close encounters
   - A Route Ribbon appears indicating the straight line route from the position at the start of the active time window to the position at the end of the active time window
   - The export button indicates that one vessel is selected
8. Use ctrl to select multiple vessels
   - All selected vessels and tracks are highlighted
   - The count of selected vessels is indicated on the export button
9. Click on the Export button in the top right
   - The export dialog appears with options to export all or just the selected vessels
10. Click on the Close Encounters button
    - Close encounter pop-ups for all vessels are toggled on and off
11. Click on the Route Ribbons button
    - Route ribbons for all vessels are toggled on and off
12. Click on the Fetch Vessels button
    - Vessels with tracks in the visible map region and time window are retrieved and replace the existing vessels
13. Click on the Timeline button
    - The timeline widget is opened using the same vessels as those on the map with a filter set to the current map time and region
14. Select a vessel and click on the View Fit button
    - The map view will adjust to focus on the selected vessel's tracks
15. Click on the Layers button

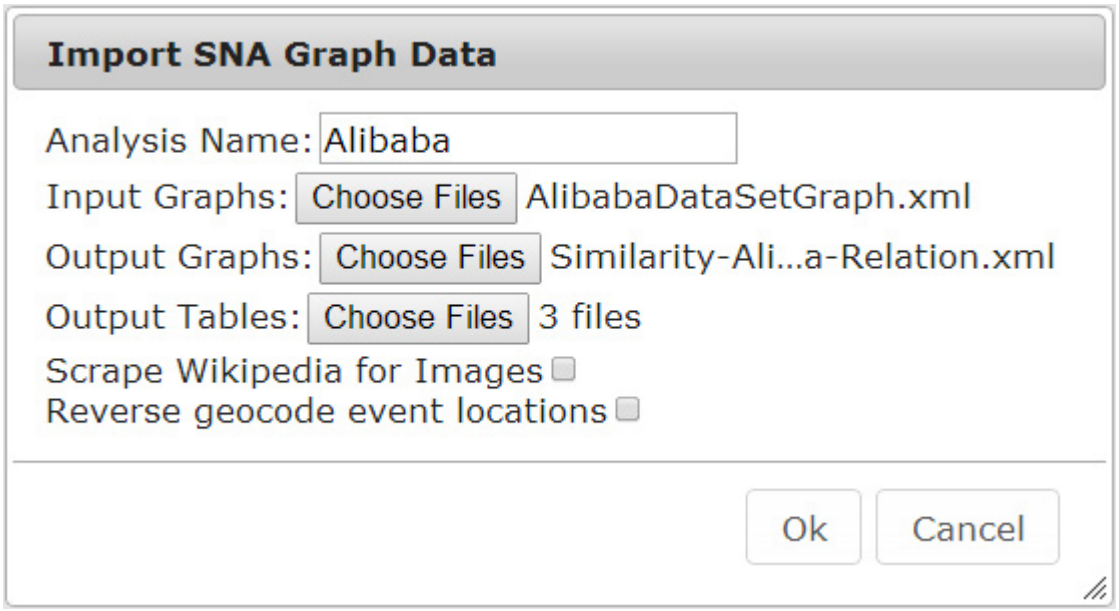
The layers dialogue appears allowing the visibility of map layers to be toggled

6.7 Standalone

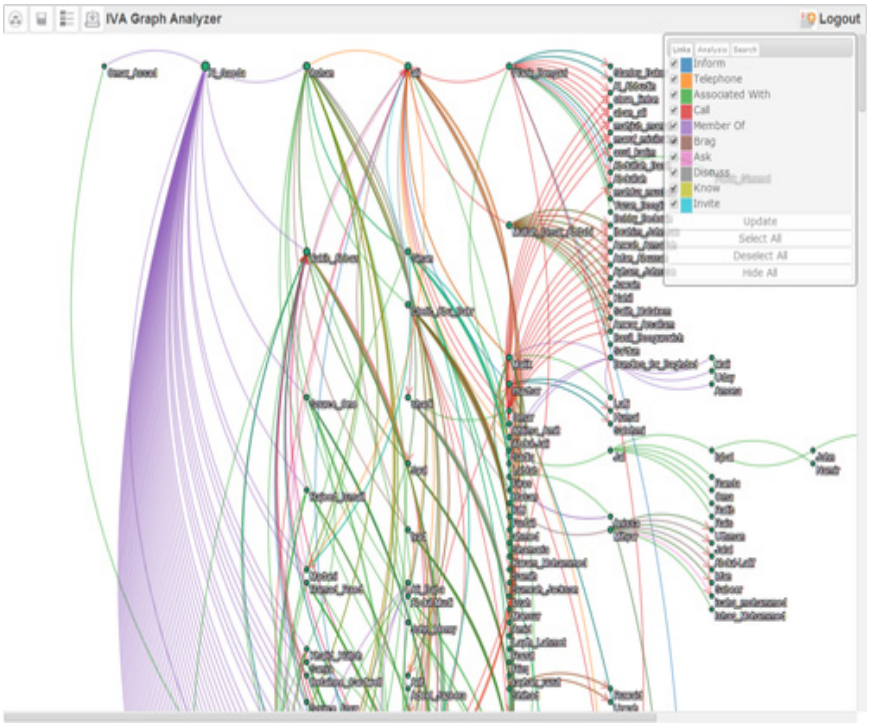
1. Connect to http://<browser_address>/MVAP using Chrome or IE9.
   - The MVAP launch page should appear with buttons to launch each stand alone widget
2. Click Magnets Grid
   - A blank instance of standalone Magnets Grid should appear
3. Drag a .csv file with a header row including “NAME” and “ID” columns onto the page
   - Dust should appear for each data row plus magnets for numeric columns
4. Add some magnets, set dust color, etc. and then click the save button in the top left
   - Bookmarking the page or refreshing should maintain the state of the session

6.8 Graph Analyzer

1. Connect to http://<browser_address>/MVAP using Chrome or IE9.
   - The MVAP launch page should appear with buttons to launch each stand alone widget
2. Click Graph Analyzer
   - A blank instance of standalone Graph Analyzer should appear
3. Click the “Import Graph Data” button to bring up the graph import dialog box
4. Enter in the “Analysis Name” field, “Alibaba”
5. In the “Input Graphs” row, click “Choose Files” to bring up a system dialog
   a. Navigate to the file “AlibabaDataSetGraph.xml” and click Ok
6. In the “Output Graphs” row, click “Choose Files” to bring up a system dialog
   a. Navigate to the file “Similarity-Alibaba-Relation.xml” and click Ok
7. In the “Output Tables” row, click “Choose Files” to bring up a system dialog
   a. Holding shift, select the following three files:
      i. SimilarityGraph-AttributeSimilarity-Alibaba-Relation.xml
      ii. SimilarityGraph-NodeSimilarity-Alibaba-Relation.xml
      iii. SimilarityGraph-SingleSimilarity-Alibaba-Relation.xml
8. Click Ok on the graph import dialog box



9. You should now see a dialog that says "Importing Graph Data" with a spinner. Graph import takes ~30 seconds. When the dialog goes away, we should now see the input graph for the Alibaba Dataset



10. To filter down the graph, click "Deselect All" in the interactive Link Legend. This deselects all link types in the legend. Now click the checkbox beside "Member Of" and click "Update"

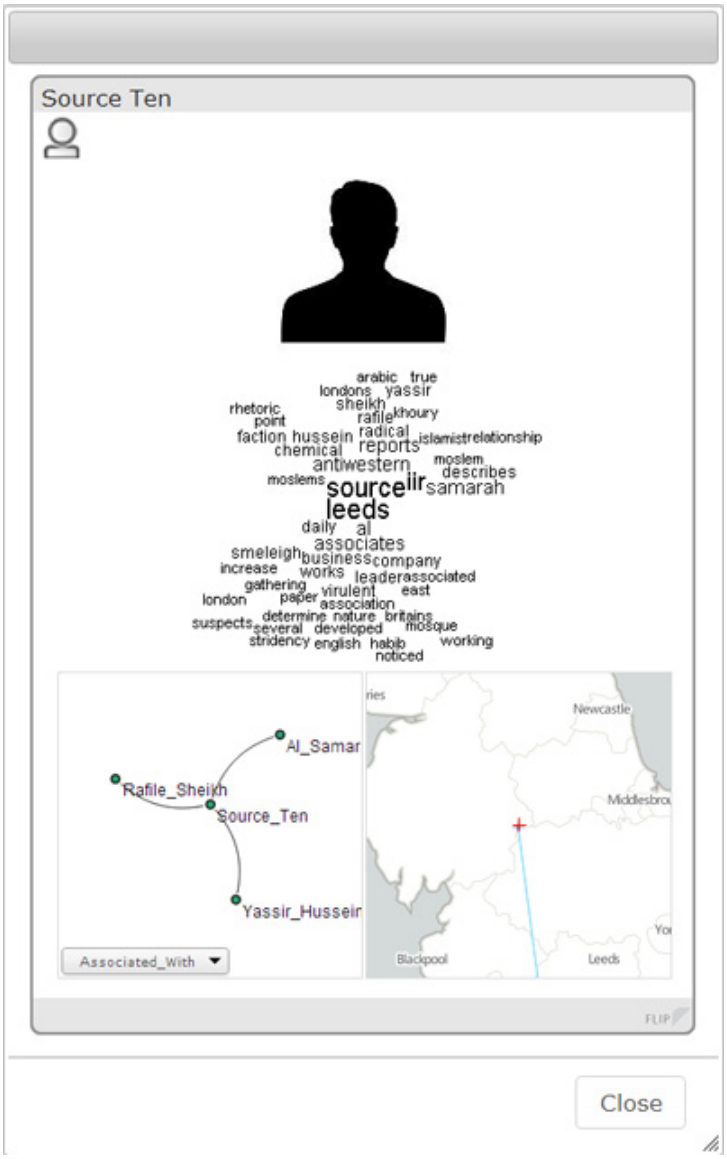
11. You should now see all nodes that have the relation “Member Of” with “Al Qaeda” in the second column. To focus that node, right click it to bring up a context menu and click “focus”. This will move “Al Qaeda” to the top left corner of the graph

12. Mouse-over “Al Qaeda” to highlight all links and destination nodes it is connected to. You will also see a tooltip displaying the name and the number of links for the node

13. Locate the node “Source_10” by clicking the “Search” tab in the interactive legend

    a. Enter the string “Source” and press enter. You should now see a node highlighted red that contains the string source. Press enter a few more times until the node “Source_Ten” is highlighted

14. Double click the node “Source_10” to bring up the summary card for this archive



15. In the Graph Inset in the bottom left of the card, you can see all nodes that Source_Ten is connected to by the relation “Associated With”

16. Click the dropdown to bring up a list of all relations this node has and click “Member Of” to see which organizations this node is a member of

17. Click “Close” on the Summary Card Dialog and click the button for the Analysis Set Manager in the top left of the main Graph Analyzer screen

18. Click the button for the Analysis Set Manager in the top left of the screen.

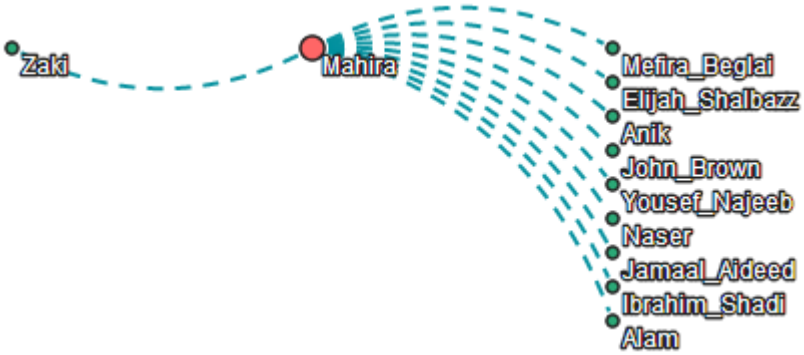
19. You will see an Analysis Set name “Alibaba”. Click + to open its children.

20. Click the Analysis Set named “Alibaba-Relation” to highlight it and click the Graph Analyzer button on the top right of the Analysis Set Manager Toolbar.

21. You should see a similar graph as before. Click “Deselect All” to remove all relational link types in the link legend. Now click the “Analysis” tab to view any Analytic Link Types this graph contains. Select “REACHABILITY” and click “Update”

22. To hide nodes that are not connected to other nodes based on this filter, click the “Hide Disconnected Nodes” button

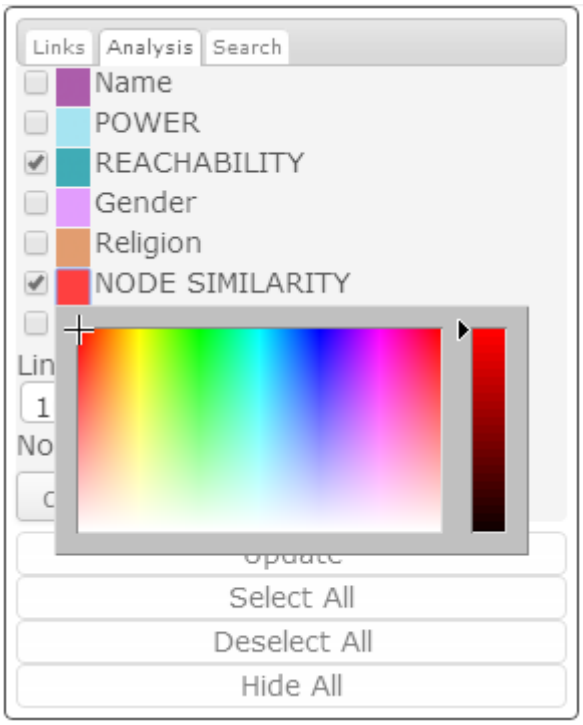
23. You should see a graph similar to the one below with the similarity node “Mahira” highlighted red



24. Hover over the links to get a tooltip and the value for the analysis metric that link is based on

25. Check the “NODE SIMILARITY” box in the legend and click update

26. If the color is clashing with the background, it can be changed by click the icon for the color in the legend next to the checkbox. Change it to a red color



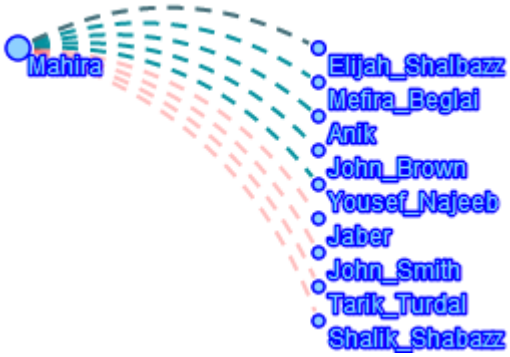
27. Change the "Link Threshold" value to 15 and click update to see the top 15 Analytic Links of type "NODE SIMILARITY" and "REACHABILITY" for the similarity node

28. Right click "Mahira" and click focus to bring the node to the top right. Click "Save" on the toolbar to save the workspace state

29. Refresh the browser and you should see the identical graph (color palette, positions, etc) that you saw before

30. Change the link threshold to 5 and click update

31. Click and drag to marquee select all visible nodes. They will be highlighted blue when they are selected



32. Click the “Export to Analysis Set” button in the top right corner of the Graph Analyzer toolbar. You should see that you have 10 selected nodes

33. Create a new analysis set called “Similar and Reachable Alibabas” and make sure “Export Selected” is checked. Click Ok to export selected nodes to a new analysis set

34. Go back to the Analysis Set Manager and you should see a new set called “Similar and Reachable Alibabas”. Click to highlight and open it with the Graph Analyzer

35. Click the “Analysis” tab and check “REACHABLITY” and “NODE SIMILARITY” and click update to view a similar graph that you saw from the previous step

36. To view relations of a specific type, we can right click a node and mouse over “Related Links” to see a list of all link types for this analysis

a. Right click “Merifa_Beglai” to bring up a context menu

b. Mouse over “Related Links” to expand a list of links for this analysis

c. Click “Member Of” to see which organizations this node is a member of

d. See the node “Hezbollah” appear with a link to “Merifa_Beglai”

References

[1] Michael Davenport, “Literature and Product Review of Visual Analytics for Maritime Awareness,” DRDC-Valcartier-CR 2009-228, October 2009

[2] Michael Davenport, “Opportunities for Applying Visual Analytics to Maritime Domain Awareness,” DRDC-Valcartier-CR 2009-227, October 2009

[3] Davenport, M., Lavigne, V. and Gouin D., “Design of a Maritime Domain Awareness Visual Analytics Prototype (DMVAP), Task 1: Design Study”, Salience Analytics Inc, DRDC Valcartier CR 2011-221, July 2011.

[4] Jasmin Bouchard and Michael Davenport, “Design of a Maritime Domain Awareness Visual Analytics Prototype (DMVAP), Task 3: Preliminary Functional and Architectural Specifications,” DRDC Valcartier Report CR 2011-223, July 2011.

[5] Michael Davenport, “Design of a Maritime Domain Awareness Visual Analytics Prototype (DMVAP), Task 4: Scenario and Dataset Specification,” DRDC Valcartier Report CR 2011-224, July 2011

[6] National Security Presidential Directive, Maritime Security Policy, NSPD-41/HSPD-13 (2004), www.fas.org/irp/offdocs/nspd/nspd41.pdf.

[7] Thomas, J., and Cook, K.A., “Illuminating the Path, the Research and Development Agenda for Visual Analytics”, NVAC, 2005. http://nvac.pnl.gov/agenda.stm

[8] Ji Soo Yi, Rachel Melton Ponder, John Stasko et al., "Dust & magnet: InteractiveVisualization for Everyday Data," Georgia Tech, ftp://ftp.cc.gatech.edu/pub/people/stasko/movies/dnm.mov, 2011.

[9] DRDC Valcartier, “Vessels of Interest Scenario”, file Vessels of Interest.com, 2012

[10] V. Lavigne. “ARP 11jm – Maritime Domain Analysis through Collaboration and Interactive Visualization,” PowerPoint slides, August 2012

Bibliography

Eccles, R., T. Kapler, R. Harper, W. Wright, Stories in GeoTime, Winner Best Paper, IEEE Visual Analytics Science and Technology (VAST) 2007.

Hugues Demers, Yannick Allard, Mike Davenport, and Irène Abi-Zeid, "Enhancing Reasoning Capabilities for the Multi-Intelligence Tools Suite – Maritime (MITS-M): Final Report," OODA Document 10-07R, June 2010.

Jean Roy and Michael Davenport, "Categorization of Maritime Anomalies for Notification and Alerting Purposes," NATO Workshop on Data Fusion and Anomaly Detection for Maritime Situational Awareness, NURC, La Spezia, 15-17 September 2009.

Jean Roy and Michael Davenport, "Exploitation of Maritime Domain Ontologies for Anomaly Detection and Threat Analysis," NURC Waterside Security Conference, Marina di Carrera Italy, November, 2010.

Jonker, J. and W. Wright, "Visualization and Comprehension," Chapter in *Estimating Impact - A Handbook of Computational Methods and Models for Anticipating Economic, Social, Political and Security Effects in International Interventions*, A. Kott, G. Citrenbaum Editors, Springer, 2010.

Kapler, T., R. Eccles, R. Harper and W. Wright, Configurable Spaces: Temporal Analysis in Diagrammatic Contexts, IEEE Visual Analytics Science and Technology VAST 2008.

Kristensson, P., et al, An Evaluation of Space Time Cube Representation of Spatiotemporal Patterns, IEEE Transactions on Visualization and Computer Graphics, Vol 15, No 4, July 2009.

MacInnes, J., S. Santosa and W. Wright, Visual Classification: Expert Knowledge Guides Machine Learning, Special Issue on Knowledge Assisted Visualization, IEEE Computer Graphics and Applications, January, 2010.

MacInnes, J., S. Santosa, N. Kronenfeld, J. McCuaig, and W. Wright, nAble Adaptive Scaffolding Agent - Intelligent Support for Novices, IEEE/ACM international Conference on Web Intelligence and Intelligent Agent Technology, 2008.

Michael Davenport, "CKEF Scenarios," MacDonald Dettwiler report CKEF-RP-52-6641 for DRDC Valcartier, 3 March 2008

Michael Davenport, "Enhancing Reasoning Capabilities for the Multi-Intelligence Tools Suite - Maritime (MITS-M) Task 3: Maritime Domain Ontology Improvements," Analytics Report SAI-10-03R for DRDC Valcartier, June 2010.

Michael Davenport, "Kinematic behaviour anomaly detection (KBAD) Final Report," DRDC CORA report KBAD-RP-52-6615, 2008.

Michael Davenport, "Knowledge Acquisition Sessions for Maritime Anomaly Detection. Volume 2: Anomaly and Threat Tables," Salience Analytics Report SAI-09-02R, DRDC-Valcartier, March 2009

Michael Davenport, “Knowledge management for maritime ISR”, presented at the Knowledge Management Information Exchange Workshop, Esquimalt Naval Base, February 2003

Michael Davenport, “Maritime Anomaly Detection Workshop Report and Analysis,” DRDC Valcartier CR 2008-275, February 2008

Michael Davenport, “Maritime Domain Awareness Knowledge Management Requirements,” DRDC Valcartier CR 2007-174, 17 June 2007

Michael Davenport, “Maritime surveillance — the future”, presented to the Armed Forced Communication Electronics Association (AFCEA), Ottawa, February 1, 2005

Michael Davenport, “Specification of the ARMAD Maritime Domain Ontology,” Salience Analytics Report SAI-09-03R, DRDC-Valcartier, May 2009

Michael Davenport and Gavin Ha, “Maritime Domain Awareness Knowledge Assets Inventory,” DRDC Valcartier CR 2007-173, 17 June 2007

Michael Davenport and Arthur Cole, “Visions of a Future Maritime Picture,” DRDC-Atlantic, CR-2006-038, March 2006

Michael Davenport and Chris Risley, “Information Visualization: the State of the Art for Maritime Domain Awareness,” Defence Research and Development Canada — Atlantic. CR 2006-122, http://pubs.drdc-rddc.gc.ca, July 17 2006

Michael Davenport and Grant Sullivan, “Study of Maritime Forces Atlantic data fusion”, DRDC-Valcartier DREV-CR-939, 940 - Vol 1&2, June 1997

Michael Davenport and Hugues Demers, “Automated Reasoning for Maritime Anomaly Detection (ARMAD), Final Report,” Ooda Technologies Inc. report Ooda-09-R0012, for DRDC-Valcartier, August 31, 2009

Michael Davenport and Peter Sarunic, “Aurora Spotlight SAR information fusion”, MacDonald Dettwiler report RX-RP-50-8335 for CF PMO Aurora, June 1998

Michael Davenport and Richard Brown, “Coalition Battle Management Language (C-BML) Study in Support of Maritime Domain Awareness,” DRDC document number not yet released, MacDonald Dettwiler report RX-RP-52-5418, March 2007

Michael Davenport, Eric Widdis, Curtis Somers, Tim Hammond, “Automatic Identification System (AIS) Spoofing,” DRDC-Atlantic, TM-2005-272, November 2005

Michael Davenport, Eric Widdis, and Lt(N) John Rafuse, “RMP baseline update”, Vol 1-3, DRDC-Atlantic TR 2005 293, TR 2005 294, TR 2005 295, March 2005

Michael Davenport, Rick Gerbrecht, James Kraft, and Greg Aikins, “Knowledge Acquisition Sessions for Maritime Anomaly Detection. Volume 1: Methodology and Interview Data,” Salience Analytics Report SAI-09-01R, DRDC-Valcartier, March 2009

Niels Willems, Huub van de Wetering, and Jarke J. van Wijk, "Visualization of vessel movements," *Eurographics/ IEEE-VGTC Symposium on Visualization*, http://www.win.tue.nl/~cwillems/public/eurovis09.pdf, 2009.

Oculus Info, "nSpace and GeoTime; VAST 2007 Contest Submission," VAST Symposium, http://vac.nist.gov/2007/submissions/OculusInfo-nSpaceAndGeoTime/index.html, 2007.

Per Ola Kristensson, Nils Dahlback, Daniel Anundi, Marius Björnstad, Hanna Gillberg, Jonas Haraldsson, Ingrid Mårtensson, Mathias Nordvall, and Josefine Ståhl, "Representation of Spatiotemporal Patterns," IEEE Trans Vis Comp Graph, Vol 15, No 4, Jul 2009.

Proulx, P., L. Chien, R. Harper, D. Schroh, T. Kapler, D. Jonker and W. Wright, nSpace and GeoTime - VAST 2006 Case Study, IEEE Computer Graphics and Applications, Sept, 2007.

Santosa, S., J. McCuaig, J. MacInnes, N. Kronenfeld and W. Wright, Anatomy of an Adaptive Scaffold for Analysts, European Conference on Cognitive Ergonomics, 2009.

Schroh, D., N. Bozowsky, M. Savigny and W. Wright, "nCompass Service Oriented Architecture for Tacit Collaboration Services," Open Source Intelligence and Web Mining Symposium, International Conference on Information Visualization, 2009.

Stuart K. Card, Jock Mackinlay, Ben Shneiderman, Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann; 1999.

Wright, W., D. Schroh, P. Proulx, A. Skaburskis, and B. Cort, The Sandbox for Analysis – Concepts and Methods, ACM CHI, 2006.

List of symbols/abbreviations/acronyms/initialisms

AIS	Automatic Identification System
API	Application Programming Interface
App	Application Software, also: Widget
ARMAD	Automated Reasoning for Maritime Anomaly Detection
ARP	Applied Research Project
CAF	Canadian Armed Forces
CBSA	Canada Border Services Agency
CCG	Canadian Coast Guard
CORA	Centre for Operational Research and Analysis
DFO	Department of Fisheries and Oceans
DMVAP	Design of a Maritime Visual Analytics Prototype
DND	Department of National Defence
DRDC	Defence Research & Development Canada
GWT	Google Web Toolkit
HCI	Human Computer Interaction
I2	Intelligence and Information
ISR	Intelligence Surveillance and Reconnaissance
ISTIP	Intelligence Science and Technology Integration Platform
JSON	JavaScript Object Notation
MCTS	Marine Communications and Traffic Services
MDA	Maritime Domain Awareness
MDO	Maritime Domain Ontology
MITS	Multi-Intelligence Tool Suite
MSOC	Maritime Surveillance Operations Centre
MVAP	Maritime Visual Analytic Prototype
OWF	Ozone Widget Framework
PAVM	Prototype d'Analyse Visuelle Maritime
R&D	Research & Development
RCMP	Royal Canadian Mounted Police
RCN	Royal Canadian Navy
REST	Representational State Transfer

RJOC	Regional Joint Operations Centres
RMP	Recognized Maritime Picture
SA	Scientific Authority
SNA	Social Network Analysis
SOA	Service Oriented Architecture
TA	Technical Authority
UI	User Interface
VA	Visual Analytics
VADA	Visual Analytics based Detection of Anomalies
VAMDA	Visual Analytics for Maritime Domain Awareness
VOI	Vessel of Interest
VOiiLA	Visionary Overarching Interaction Interface Layer for Analysis
VTS	Vessel Tracking System
WAS	Widget Application Shell
XML	eXtensible Markup Language